JMLR: Workshop and Conference Proceedings 36:29–45, 2014 BIGMINE 2014

FAQ: A Framework for Fast Approximate Query Processing on Temporal Data

Udayan Khurana UDAYAN@CS.UMD.EDU
University of Maryland, College Park, MD

Srinivasan Parthasarathy SPARTHA@US.IBM.COM
IBM T.J. Watson Research Center, Yorktown Heights, NY

Deepak Turaga TURAGA@US.IBM.COM
IBM T.J. Watson Research Center, Yorktown Heights, NY

Editors: Wei Fan, Albert Bifet, Qiang Yang and Philip Yu

Abstract

Temporal queries on time evolving data are at the heart of a broad range of business and network intelligence applications ranging from consumer behavior analysis, trend analysis, temporal pattern mining, sentiment analysis on social media, cyber security, and network monitoring. In this work, we present an innovative data structure called Fast Approximate Query-able(FAQ), which provides a unified framework for temporal query processing on Big Data. FAQ uses a novel composition of data sketching, wavelet-style differencing for temporal compression, and quantization, and handles diverse kinds of queries including distinct counts, set membership, frequency estimation, top-K, p-norms, empirical entropy, and distance queries such as Histogram ℓ_p-norm distance (including Euclidean and Manhattan distance), cosine similarity, Jaccard coefficient, and rank correlation. Experiments on a real-life multi dimensional network monitoring data sets demonstrate speedups of 92x achieved by FAQ over a flat representation of data for a mixed temporal query workload.

1. Introduction

Temporal aggregation, similarity and distance queries are at the heart of several important data exploration and knowledge discovery tasks such as prediction, forecasting, classification, clustering, search and retrieval on time evolving data sets [21, 18, 24, 2]. Efficient temporal query processing is a critical factor in our ability to understand and leverage the ocean of data that is continuously generated in our interconnected environment. The process of exploratory data analysis and knowledge discovery often poses additional challenges for temporal query processing since the data scientist is interested in exploring the data through multiple lenses: what interesting behaviors and patterns emerge when the data is viewed at the granularity of a day vs week vs month? How does the data cluster when the clustering algorithm uses Manhattan vs Euclidean distance? A cardinal question that emerges here is as follows: *Are there algorithm design and data structure principles that can be leveraged for efficient temporal query processing at multiple granularities* ***and*** *are they robust and generic enough to handle a variety of aggregation, similarity and distance functions?* We answer this question in the affirmative by presenting an innovative framework

called Fast Approximate Query-able(FAQ). We begin with a few scenarios that illustrate temporal queries and subsequently describe the FAQ framework.

Illustrative Temporal Queries:

1. Consider the set of Twitter users and the subset of Tweets containing mentions of consumer products. What were the top 10 frequently mentioned products across all users belonging to a given geographical region over a given hour, day, week or month? What is the similarity score between the set of products mentioned today compared to the day before?
2. Consider two blogging communities each of which generates a time evolving collection of blogs. How divergent are the set of topics mentioned by the two communities over a given hour, day, week, or month? What is the divergence score if measured in terms of the Jaccard coefficient, Manhattan distance, or Euclidean distance?
3. Consider the set of all devices in an enterprise network and the DNS queries made by these devices. Given a subnet, what is the number of distinct domains queried by all the devices in this subnet over a second, hour, day, week, or 3-month interval? What is the empirical entropy and the second frequency moment of the query distribution?

Key Aspects of Temporal Query Processing: The aforementioned scenarios illustrate a few key aspects of the problem we address in this work, namely: (i) Entities often exhibit behavior that is sparse and stable over time: for instance, the set of DNS queries emanating from a device is determined by various applications installed on the device as well as the web browsing preferences of the owner of the device; in practice, both of these change infrequently over time. Temporal aggregation and distance queries can reap tremendous benefits in terms of efficiency if **temporal sparsity** of this kind can be exploited appropriately[1]; (ii) Temporal data could be either real-valued vectors (e.g., frequency or empirical distribution over DNS domains) or categorical (e.g., set of products mentioned); (iii) Aggregation could be over multiple time resolutions and more generally, over arbitrary time intervals; (iv) Aggregation could be across various entities each of which is a distinct source of the time evolving data (e.g., multiple devices in a subnet, each of which generates its own DNS queries); (v) Aggregation, similarity and distance functions could be of diverse types; and (vi) Exploratory data analysis applications generally tend to be tolerant of small bounded errors. Processes such as NLP, entity recognition, and sentiment extraction are often approximate and probabilistic in themselves and hence a marginal increase in error introduced by query processing for the purpose of delivering substantial performance benefits is a welcome trade-off.

Existing Approaches: The preeminent approach for answering aggregate, similarity and distance queries on time evolving data the approach of data sketching; we refer the reader to Cormode et al. [10] for a useful survey of sketching techniques. Sketching algorithms are often simple and practical to implement, and many sketches achieve significant space efficiency by constructing various succinct random estimators from the raw data and yielding a function of these estimators as the answer to the query. However the literature on sketching algorithms do not generally focus on efficiently handling multi-resolution queries posed

1. This fact is borne out by our experiments in Section 5

over arbitrary time windows, and do not provide a means for exploiting temporal sparsity for efficiently processing temporal aggregation queries. Wavelet transformation of time series data is a well-studied alternate representation that is particularly suitable for multi-resolution analysis of time-series data as observed by Chan et al. [7] and provides significant compression benefits for temporally sparse data. However, wavelets are geared towards signal reconstruction and recovery and generally do not achieve the level of compression achieved by sketching algorithms (often poly-logarithmic in the size of the original signal). Further, wavelets are geared towards real valued time series and are not directly applicable for categorical data such as time evolving sets or graphs.

The FAQ Approach: The core contribution of this work is a novel data structure for temporal aggregation, similarity, and distance query processing called Fast Approximate Query-able index (FAQ). FAQ uses Wavelet-style differencing techniques to convert any given data sketching scheme into a multi-resolution range-tree that is suitable for temporal query processing. For real-valued histograms and sketches, FAQ introduces a novel geometric quantization scheme which yields further benefits in terms of compression. The flexible composition of these algorithmic techniques enables FAQ to handle diverse types of temporal queries with a high degree of accuracy while achieving substantial speedups on real-world data sets.

Specific Contributions: The specific contributions of this work are as follows.

1. **Multi-resolution:** We introduce an innovative data structure called FAQ which converts any data sketching scheme into a multi-resolution range tree structure that is ideal for handling temporal aggregation, similarity, and distance queries. This structure guarantees bounded (squared-logarithmic) traversal time for any arbitrary time range over which the aggregation query is posed.
2. **Temporal Sparsity:** FAQ explicitly identifies and leverages opportunities for temporal compression in the presence of temporal sparsity. Specifically, during its construction phase, FAQ explores multiple possible data representations and chooses the locally space-optimal representation for each FAQ node / edge in order to achieve temporal compression.
3. **Geometric Quantization:** FAQ introduces a novel geometric quantization technique that can be applied to histograms and real-valued sketches which yields added benefits in terms of compression. The quantization error introduced by this encoding can be analytically upper bound; this opens up interesting optimization opportunities for the FAQ query planner to straddle across multiple versions of the FAQ data structure (each representing the same underlying data set) in order to speedily answer a given query.
4. **Generality via Flexible Composition:** The flexible composition of the aforementioned algorithmic techniques makes FAQ a highly efficient general purpose data structure for answering a wide variety of aggregation queries such as distinct counts, set membership, frequency estimation or top-K, p-norms, empirical entropy, and distance and similarity queries such as Histogram distances (ℓ_p-distances including Euclidean and Manhattan distance), cosine similarity, Jaccard coefficient, and Rank correlation.

5. **Empirical Evaluation:** We demonstrate the benefits of FAQ in terms of accuracy and speedups for histogram aggregation and distance queries on two large real-world network data sets.

Outline: In the remainder of this paper, we survey related work in Section 2; we provide formal definitions of our data and query model in Section 3, and describe the construction and operation of FAQ in Section 4. We demonstrate the power and benefits FAQ for histogram aggregation and distance queries on large real-world network data sets in Section 5 and conclude with Section 6.

2. Related Work

2.1. Sketching

Sketching algorithms have been developed over the years for a wide variety of useful queries; refer to Cormode et al. [10] for a survey of such techniques. Specific sketches have been developed for common temporal aggregation functions such as distinct counts [12, 1, 4], set membership [14], frequency counts and top-K items [11, 8], frequency moments, p-norms, and ℓ_p-distances [15, 19], empirical entropy [20], Jaccard coefficient [5] and Cosine similarity [9]. Our work builds on top of existing research on data sketches and provides a way to leverage all of these sketches within the FAQ framework which is well-suited for multi-resolution temporal queries.

2.2. Frequency Domain Transformation

Frequency domain transformations of time series measurements underlie much of signal processing theory and applications. An overview of such transforms and specifically, their applications to time series distance metric computation is presented by Wang et al. [28]. These transforms include the Discrete Fourier, Cosine, and Wavelet transforms, Singular Value Decomposition, Polynomial approximation schemes, Symbolic Aggregate approXimation, and Indexable Piecewise Linear Approximation. This work also includes a discussion on the computation of different distance functions *between* time series fragments including the Euclidean distance, Dynamic Time Warping (DTW), distance based on Longest Common Subsequence, Edit Distance with Real Penalty, Edit Distance on Real sequence, DISSIM, Sequence Weighted Alignment model, Spatial Assembling Distance, and similarity search based on Threshold Queries (TQuEST). In a recent paper, Rakthanmanon et al. [25], show several optimizations in similarity computation, tailored for efficient approximate searching within very large time series datasets. Kim et al. [17] present a document retrieval solution based on time series similarity computation.

Multi-resolution representation as well as quantization are both canonical algorithmic techniques which underlie the FAQ data structure and which are inspired by the world of frequency domain signal transformations. However, there are three specific aspects of FAQ which go beyond these well-known signal encoding techniques: first, our focus is on efficiently processing aggregate queries, and aggregated histogram distance queries. Specifically, our queries involve aggregations over a set of multi-dimensional time series across time intervals and distance queries posed over such aggregates. While the frequency domain and wavelet transformations tend to focus on signal reconstruction, our focus is on

approximate aggregate and distance query processing; we approach this problem by *composing* wavelet-style multi-resolution transformation techniques *on top of sketching algorithms* which provide substantial dimensionality reduction for *each snapshot (or sample) in time in addition to compression across time*. Second, FAQ is a generic data structure that handles not only continuous or real-valued multi-dimensional time-series data, but *also discrete data such as time evolving sets and temporal aggregation queries on them such as distinct count, membership, and set similarity*. Third, the nature of quantization introduced by FAQ ensures that the number of bits needed for signal encoding is logarithmic in its range (and not linear), and *simultaneously* provide a way to probabilistically upper bound the error introduced by this encoding. We are not aware of such geometric quantization schemes employed in the context of query processing.

2.3. Timeseries Indexing

There has been a considerable amount of work on timeseries indexing in the last few years, mostly aimed at efficient specific mining and searching applications. Most of the work considers search across a large number (millions) of 1-D time series. For instance, Niennattrakul et al. [23] present an index for searching similar time series using DTW. It indexes the sequences in database using groups and when a query is issued, relevant groups are filtered based upon lower bound on distance. iSAX [27] present a novel multi-resolution symbolic representation using a tree based index structure to facilitate fast exact search on millions of one dimensional time series of small lengths (typically 16). iSAX2+ [6] extends the indexing to beyond 1 billion time series. Reeves et al. [26] use multi-scale analysis to decompose time series and obtain sparse representations in different domains, such as the time and frequency domain. These sparse representations are stored compactly on disk and queries such as trend, histogram and correlations can be answered directly without reconstructing the raw data. Mueen et al. [22] present an efficient way to compute all-pair correlations in a warehouse of tens of thousands of time-series, based on Discrete Fourier Transforms. In [13], Gandhi et al. present a framework for on-line approximation of one-dimensional time series. DeltaGraph [16] is a hierarchical index for large temporal graphs. It uses graph deltas for compact disk storage and provides a near I/O-optimal snapshot reconstruction. Wang et al. [28] present an extensive experimental study by re-implementing eight different time series representations, nine similarity measures and testing their effectiveness on thirty-eight different time series datasets from a wide variety of application domains.

FAQ is different from aforementioned indexing techniques as our data structure is intended to be generic. FAQ handles i) a wide variety of aggregation and distance queries; ii) multiple multi-dimensional time series as part of aggregation; iii) handles both real valued and categorical time evolving data and queries on them. Our data structure can be used in a database for general purpose time series applications involving fetching and similarity computation. While it is very useful to build targeted indexes with a focus on specific tasks, our data structure focuses on generic query optimization techniques that can be plugged into a variety of exploratory data analysis and data mining tasks.

3. Definitions

3.1. Data and Query Model

We consider data in the form of discrete **time-evolving histograms** defined over a set of source and target entities U and V respectively. Time is discrete, and for every time slot in $[0, 1, 2, \ldots]$, for each source entity $u \in U$, we are given a histogram $\sigma_t(u)$ which is a vector of counts defined over the target entities V. Given a histogram σ, we will let $\sigma[j]$ denote the value of its j^{th} component, and $\Omega(\sigma)$ its support, i.e., the set of all non-zero components of σ. Below, we provide the formal description of the data sets used as illustrative examples in Section 1.

1. U and V represent the set of Twitter users and consumer products, respectively. $\sigma_t(u)[j]$ is the number of mentions of product j by user u during the hour t.
2. U and V represent the set of all blogs and topics, respectively. $\sigma_t(u)[j]$ is the number of mentions of topic j in blog u during the hour t.
3. U and V represent the set of all devices in an enterprise network and all DNS domains on the Internet, respectively. $\sigma_t(u)[j]$ is the number of queries made for domain j by device u during the second t.

For a given subset of source entities $A \subseteq U$, and for any time range $[t_1, \ldots, t_2]$, the **aggregate histogram** $\sigma_{[t_1,\ldots,t_2]}(A)$ is defined as the histogram obtained through the component-wise addition of the $|A| \times (t_2 - t_1 + 1)$ base histograms. The **aggregation queries** we deal with are of the general form: $f(\sigma_{[t_1,\ldots,t_2]}(A))$ where f is a histogram function such as entropy or the ℓ_2-norm. The **similarity and distance queries** we deal with are of the general form: $g(\sigma_{[t_1,\ldots,t_2]}(A), \sigma_{[t_3,\ldots,t_4]}(B))$, where g is a similarity function such as the Jaccard Coefficient or a distance function such as the Euclidean distance. We are interested in efficiently answering a wide variety of temporal aggregation, similarity and distance queries, including but not limited to distinct counts, set membership, frequency counts, top-K items or heavy-hitters, frequency moments, p-norms, ℓ_p-distances, empirical entropy, Jaccard coefficient, rank correlation, and Cosine similarity.

3.2. Data Sketch

Sketches are succinct data structures that can be used to approximately and efficiently answer a variety of aggregation queries. A data structure $\Gamma(\sigma)$ computed from histogram σ is a sketch iff there is a space-efficient combining algorithm Ψ such that for every two histograms σ_1 and σ_2, we have $\Psi(\Gamma(\sigma_1), \Gamma(\sigma_2)) = \Gamma(\sigma_1 \circ \sigma_2)$, where $\sigma_1 \circ \sigma_2$ is the aggregation of the two histograms[2]. A linear sketch is a linear function of its histogram input σ. Note that Ψ is simply the addition operation in the appropriate vector space in this special case. A real sketch of size n is a linear sketch which takes values from the vector space $\mathbb{R}^n$.

2. By space efficient, in this paper, we will mean linear in the size of the inputs to Ψ; For instance, when we deal with random projection sketches such as those employed for p-norms and ℓ_p-distances, $\circ$ and Ψ are both vector additions, the former in the histogram domain and the latter in the sketch domain; when we deal with sketches that are bitmaps as in a Bloom filter, $\circ$ represents set union while Ψ is the bit-wise OR operation

4. The FAQ Data Structure

The FAQ data structure is a composition of three distinct algorithmic principles, namely i) range-tree organization, ii) sparse representation and iii) geometric quantization. These algorithmic principles can be applied purely in the histogram domain, or in the sketch domain, or a hybrid of the two. We first describe these algorithmic principles in the following three subsections and subsequently describe how they can be configured together in multiple way within the architecture of our current prototype.

4.1. Range-tree Organization

Given a source entity u, consider the set of all sketch values associated with this entity: i.e., $\forall t,\ \Gamma(\sigma(u)[t])$. FAQ organizes these sketches for u in the form of a binary[3] range-tree in order to expedite temporal range queries for this source. This concept is illustrated in Figure 1(a). The root of this binary tree contains the aggregate sketch for u over all time-slots, and the leaf nodes contain the sketches for individual time slots. Consider an internal node of this tree which spans a set of contiguous time slots w. The left and right children of this node span the left and right halves[4] of the time range w respectively. The Lemma below follows from the standard analysis of range trees.

Lemma 1 *The aggregated sketch for any arbitrary time range $[t_1, t_2)$ can be computed by combining the sketch values of at most $O(\log n)$ internal and leaf nodes. For any internal node, the sketch value can be computed by traversing a path of at most $O(\log n)$.*

4.2. Sparse Representation

In this subsection, we describe how temporal sparsity can be opportunistically leveraged during the construction of a hybrid FAQ structure that contains a combination of sketch values, raw histogram values, sketch domain and histogram domain differences. Time evolving histograms can manifest sparsity in a variety of different ways: (i) a histogram for a particular time-range could be low dimensional: i.e., only a small number of histogram components are non-zero, or (ii) the histograms could display temporal continuity: consider histograms $\sigma_{t_0}(u)$ and $\sigma_{t_0+1}(u)$: the histogram difference $\sigma_{t_0+1}(u) - \sigma_{t_0}(u)$ could be a sparse vector, or iii) the histogram of a parent node could be a scaled up version of one of its children plus a sparse perturbation, or iv) sparsity could manifest in the sketch domain: i.e., $\Gamma(\sigma_{t_0}(u))$, or $\Gamma(\sigma_{t_0+1}(u))$ or $\Gamma(\sigma_{t_0+1}(u)) - \Gamma(\sigma_{t_0}(u))$ is a sparse vector. Table 1 presents the sketch domain differences for a few well-studied sketches. The FAQ data structure is designed to exploit all of the above opportunities for sparse representation.

Let us now consider the construction of FAQ for a specific source entity u. FAQ node and edge values are created for u by prioritizing the lower levels: i.e., nodes or edges farther from the root are processed prior to the nodes or edges closer to the root; within a given level, FAQ processes nodes from left to right. While processing any node, FAQ considers the following four options: (1) storing the raw histogram or sketch value for this node; (2)

3. The range-tree may have a higher arity than two. The binary case is used for simplicity of illustration.
4. This description assumes that the total number of time slots is a power of two. This assumption can be trivially realized by rounding up the number of slots to the nearest power of two, and assigning zero-valued histograms to the artificially added slots

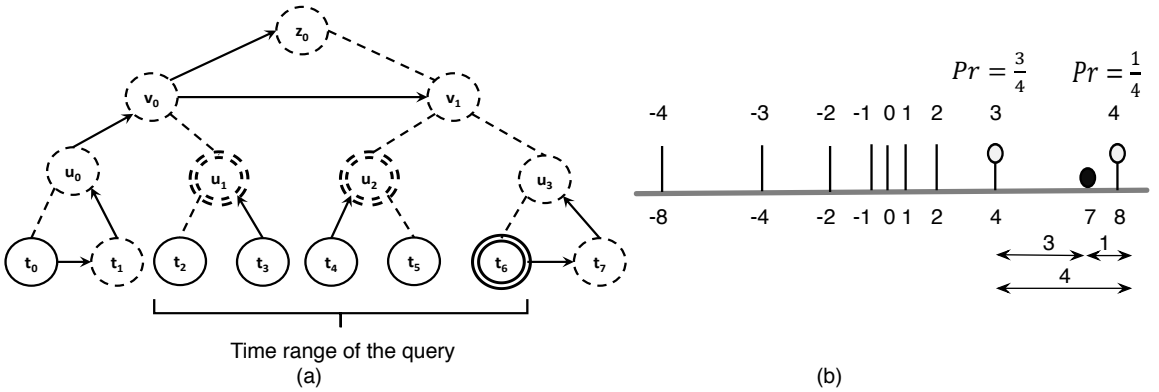

Figure 1: (a) **FAQ Range Tree:** The histogram and / or sketch values for a source at the level of individual time units as well as aggregated time intervals represented as a binary range-tree. Given an aggregate query involving time window $[t_2, \ldots, t_6]$, FAQ needs to aggregate values at nodes u_1, u_2, and t_6 in order to arrive at the result. In this illustrated example, the dotted nodes or edges do not have any values associated with them, while solid nodes and edges do. Computing the value at node z_0 will involve traversing v_1, v_0, u_0, t_1 and t_0 and aggregating the values found along the nodes *and* edges in this traversal; (b) **Geometric Quantization**: Illustration of geometric quantization with $\alpha = \epsilon = 1$. Values in the bucket $(-1, 1)$ are rounded to zero. If a value p falls in the bucket $(2^i, 2^{i+1})$, we store $i+1$ with probability $2 - \frac{p}{2^i}$ and store $i+2$ with probability $\frac{p}{2^i} - 1$. For instance, if $p = 7$, then p is rounded up to 8 and the value 4 is stored with probability $\frac{3}{4}$, and it is rounded down to 4 and the value 3 is stored with the complementary probability $\frac{1}{4}$. Negative values are treated analogously.

and (3) storing the difference with respect to its left or right child along the left or right edges down the tree, respectively and (4) if this node is a right child, storing the difference with respect to its left sibling along the left edge. In all of the above cases the raw values or differences could be either in the histogram domain or the sketch domain. While processing a node, FAQ greedily chooses that option among the above eight choices which minimizes the storage needed at this node. The notion of storing both node *and* edge-wise values is illustrated in Figure1(a).[5]

4.3. Geometric Quantization

The final layer of optimization that FAQ employs for real sketches is geometric quantization (illustrated in Figure 1(b)). The quantization logic is parameterized by two small positive constants α and ϵ. The positive real axis is divided into buckets of geometrically increasing lengths: $[0, \alpha), [\alpha, \alpha(1+\epsilon), [\alpha(1+\epsilon), \alpha(1+\epsilon)^2), \ldots$. We designate the bucket $[0, \alpha)$ as the positive zero bucket. Consider a real sketch and one of its components with a non-negative value p. If p falls in the bucket $[a, b)$, it is rounded up to value b with probability $\frac{p-a}{b-a}$ and rounded down to a with the complementary probability. Components rounded to zero

5. The FAQ structure is cyclic due to edges between siblings and hence not a tree in the strict graph theoretic sense of the term. We use the term tree due to FAQ's structural similarity to a range tree.

Query type	**Data sketch**	**Native representation**	**FAQ Representation**
Distinct Count	Flajolet-Martin Sketch [12]	$\Gamma(\sigma) = \gamma_1, \ldots, \gamma_k$ each of which is an integer random variable	$\Omega(\sigma_1), \Omega(\sigma_2), \Gamma(\sigma_1), \Gamma(\sigma_2)$, $\Omega(\sigma_1) \setminus \Omega(\sigma_2)$, $\Omega(\sigma_2) \setminus \Omega(\sigma_1)$, non-zero components of $\Gamma(\sigma_1) - \Gamma(\sigma_2)$
Frequency Estimation and Top-K	CCFC Sketch [8]	$\Gamma(\sigma) = \gamma_1, \ldots, \gamma_k$ each of which is a real random variable	Non-zero components of σ_1, σ_2, $\sigma_1 - \sigma_2$, $\Gamma(\sigma_1)$, $\Gamma(\sigma_2)$, $\Gamma(\sigma_1 - \sigma_2)$
	Count Min-Sketch [11]		
Frequency moments	Random projection sketches [9, 1, 15, 20, 19]		
p-norms			
Histogram distances			
Empirical entropy			
Cosine similarity			
Jaccard Coefficient	Min-wise independent permutation [5]	$\Gamma(\sigma) = \{\gamma_1, \ldots, \gamma_k\}$ all of which are elements from $\Omega(\sigma)$	$\Omega(\sigma_1)$, $\Omega(\sigma_2)$, $\Gamma(\sigma_1) \setminus \Gamma(\sigma_2)$, $\Gamma(\sigma_2) \setminus \Gamma(\sigma_1)$
Rank Correlation	Correlation Sketch [3]		
Set membership	Dynamic Bloom Filters [14]	$\Gamma(\sigma) = [\gamma_1, \ldots, \gamma_k]$, a bit vector	$\Omega(\sigma_1)$, $\Omega(\sigma_2)$, $\Gamma(\sigma_1)$, $\Gamma(\sigma_2)$, $\Omega(\sigma_1) \setminus \Omega(\sigma_2)$, $\Omega(\sigma_2) \setminus \Omega(\sigma_1)$, $\Gamma(\Omega(\sigma_1) \setminus \Omega(\sigma_2))$, $\Gamma(\Omega(\sigma_2) \setminus \Omega(\sigma_1))$

Table 1: Sparse FAQ Representation options for various Sketches. The same native representation often underlies multiple different sketches (for e.g., random projection sketches for multiple applications all produce a random real vector). In the last column, σ_1, and σ_2 could correspond to either parent and child nodes or left and right siblings in the FAQ range tree. For instance, in the case a random projection sketch, we might store the sketch of the parent, and the sketch domain *or* histogram domain difference between the parent and a child; the latter option requires that the histogram domain difference be sketched and added to the parent to derive the child's sketch value while answering a query. The various options and the space-optimal selection among these options are described in Section 4.2.

are discarded; we store the exponent $i + 1$ whenever a component is rounded to the value $\alpha(1 + \epsilon)^i$. Negative components in the sketch are treated analogously.

The key benefit of the geometric quantization scheme is the reduction in the storage (number of bits) needed for individual sketch components: for instance, with $\alpha = 1$ and $\epsilon = 0.01$, we can store any sketch component in the range $(-4.006601e + 141, 4.006601e + 141)$ using a 16-bit signed integer value as opposed to a floating point value. Of course, quantization also introduces additional errors which we analyze below.

FAQ answers a query by creating an aggregated sketch that is composed from multiple constituent sketches. Consider an aggregated sketch and one of its component values x which is derived by summing the corresponding components in the constituent sketches: i.e., $x = \sum_i x_i^{pz} + \sum_j x_j^{nz} + \sum_k x_k^p + \sum_l x_l^n$. In the preceding expression, we have grouped together the constituent values that make up x into four groups, the values that fall in the positive zero bucket (x^{pz}), the negative zero bucket (x^{nz}), positive non-zero buckets (x^p), and negative non-zero buckets (x^n). Now consider their quantized analogues: $\hat{x} = \sum_i \hat{x}_i^{pz} + \sum_j \hat{x}_j^{nz} + \sum_k \hat{x}_k^p + \sum_l \hat{x}_l^n$. It is easy to verify the following lemmas.

Lemma 2 *The quantized sums $\sum_k \hat{x}_k^p$ and $\sum_k \hat{x}_k^n$ are at most a factor of $(1+\epsilon)$ away from their un-quantized counterparts $\sum_k x_k^p$ and $\sum_k x_k^n$ respectively.*

Lemma 3 *The expected value of a quantized value equals its original un-quantized value.*

Lemmas 2 and 3 allow us to use large deviation bounds to probabilistically bound the maximum error introduced via quantization: if this error estimate is within acceptable limits for a given query, we use the values yielded by the quantized sketch; if not, we recompute the answer to the query using the un-quantized FAQ.

4.4. FAQ Architecture

The architecture of our current FAQ prototype is illustrated in Figure 2(a). The main components are as follows:

- *Index* consists of multiple representations of the raw data, each of which imposes its own storage requirement, provides accuracy guarantees, and performance in terms of query response times. For instance, FAQ used for the computation of histogram distances using random projection sketches could have three distinct representations: (i) A version based on pure histograms without sketching or quantization, and hence yielding error-free results, (ii) A version based on a hybrid of sketches and histograms, and (iii) A version based on a hybrid of quantized sketches, and histograms. In the current prototype, the construction parameters of the index are specified manually.
- *Query Planner* is responsible for selecting the appropriate representation in order to answer a given query. The optimal representation answers a query within the required bound on error, in the shortest possible time amongst all other representations: for instance, if the quantized version yields a (fast) result but exceeds the error bounds specified by the query, the planner might resort to the un-quantized version or the pure histogram version of FAQ. In general, the query planning component employs a variety of different optimization strategies and heuristics which beyond the scope of this paper.
- *Index Manager* handles the different representations in the index, and the computation of aggregates. It also deals with memory materialization, i.e., caching of certain parts of the range trees for speeding up the query processing.
- *Error Estimator* estimates the errors involved in a particular query evaluation. It assists the query planner in making the decisions on selecting across representation.

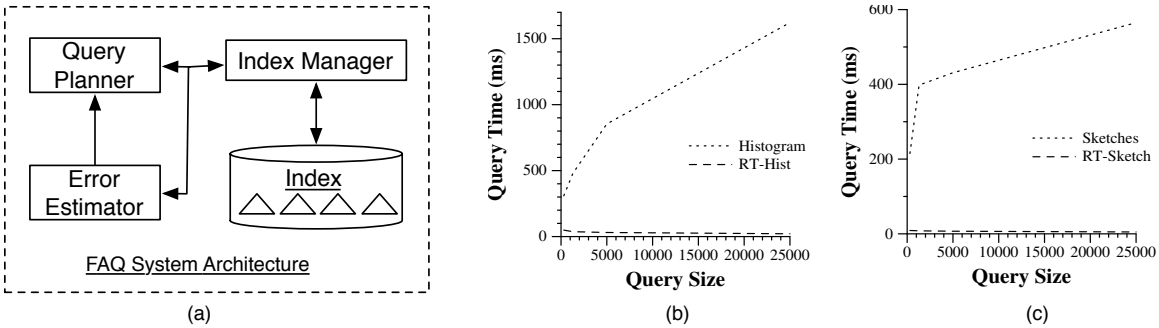

Figure 2: (a) **FAQ system architecture** highlighting the Query Planner, Index Manager, Error Estimator and the Index; Query times for (b) Range tree (Histogram based) vs. Histograms and, (c) Range tree (Sketches) vs. Sketches. It can be seen that while the query time increases linearly for histograms or sketches with increasing query size, it remains relatively constant for range tree (in fact, there is a logarithmic decrease).

5. Empirical Evaluation

In this section, we present the performance evaluation conducted on the FAQ prototype. Our intent is to demonstrate the *time-accuracy* tradeoff through the selection of different representations of timeseries, the selection of several parameters for each representation, and the overall benefits through FAQ. We focus on evaluating one kind of query in an extensive manner, in order to understand the impact of different components of FAQ. We present results from similarity computation over multi-resolution aggregation of timeseries.

5.1. Implementation and Dataset

We implemented the system in Java using KyotoCabinet[6] for disk storage. Since the histograms are sparse, we used a representation that can be thought of as a list of (dimension, value) pairs. The dimensions can be short integers or integers, while the values are integers. The sketches are represented as an array of double floating point numbers. The quantized histograms or sketches use short ints. The range tree is a homogenous structure and any node can contain either of the representations mentioned above. The experiments were based on computing euclidean distance over several temporal and entity aggregations of timeseries. The general query can be expressed as, $d(\sigma_{[t_1,\ldots,t_2]}(A), \sigma_{[t_3,\ldots,t_4]}(B))$, where $\sigma_{[t_1,\ldots,t_2]}(A)$ and $\sigma_{[t_3,\ldots,t_4]}(B)$ are aggregated histograms over time periods $[t_1,\ldots,t_2]$ and $[t_3,\ldots,t_4]$, respectively for entities A and B, respectively. We make use of the following terms while reporting the results:

- *Query Time* is the response time taken to evaluate a given distance query. This involves fetching both histograms or alternate representations (sketches) from FAQ, followed by the distance computation. Unless otherwise specified, the results are reported in milliseconds.

6. http://fallabs.com/kyotocabinet/

- *Query Size* is the number of time slots across which the histograms are aggregated in query. This could be different for the two sets of histograms, but in the our experiments, we keep it the same for both sides, unless specified otherwise.
- *Entity Count* is the number of entities across which the aggregation was performed in the query. Unless specified otherwise, A and B have the same entity count.
- *Accuracy* is defined as the percentage difference from the answer obtained while using histograms. Since we treat histograms as the base data, its accuracy is always 100%.

The following parameters influence FAQ construction and hence the size, accuracy and query time as well.

- *Dimensionality(D)* is the number of bins in the histograms. For example, it is the total number of ports in Dataset 1 (described below).
- *Sketch Size(k)* is the size of the sketch vector used in a particular representation.
- *Arity(r)* is the branching factor of range tree.

The minimum confidence was fixed at 0.95 (i.e., $error < 0.05$) for all the experiments. We evaluated FAQ for two datasets:

Dataset 1: This dataset contains one hour of anonymized traffic packets from CAIDA's equinix-sanjose monitor[7]. We processed the data to create histograms with bins consisting of the frequency destination ports from the base packet data. Each histogram represents the activity aggregated over a 40 millisecond slot. In all there are 90,000 histograms.

Dataset 2: This dataset is slight modification of the dataset 1 where all the traffic is further partitioned by source port. We used a subset of this data with the source ports 1 to 200, due to which there were 18 million histograms in all.

5.2. FAQ speedup over raw histograms

We created a FAQ structure on dataset 1 containing, (a) Raw histograms on disk, (b) Range tree with sketch size, $k = 140$ on disk, no quantization, (c) Range tree with sketch size, $k = 100$ on disk, no quantization. (d) A quantized version of range tree of sketches with sketch size, $k = 100$ in main memory, consuming 87MB.

On the other hand, we chose a representation of raw histograms, which used 87 MB of memory (assigned to an arbitrary contiguous chunk of histograms) and disk for the rest (roughly, 1.9 GB). We chose 100 queries at random with time range, w varying from 400ms to 40,000ms in the multiples of 40ms and with varying $\epsilon_{max} \in [85\%, 95\%]$, chosen at random. The average time taken for a query by FAQ was, 31ms, where as the average time taken by raw histograms was, 2940ms, which is a **speedup of 92x**.

5.3. Querying on histograms and sketches

Sketches speed-up distance computation over histograms considerably. On dataset 1, we observed a speedup by a factor of 5-40 times with an accuracy in the range of 82%-97%. In

7. http://www.caida.org/data/passive/passive_2012_dataset.xml

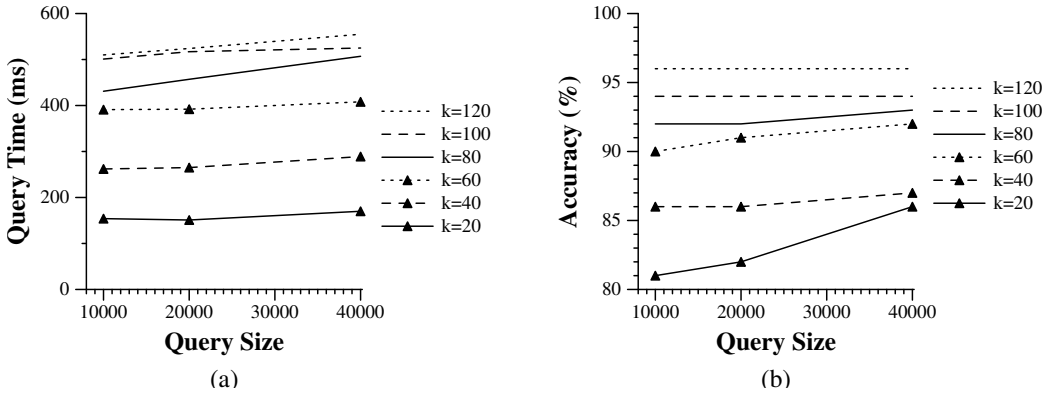

Figure 3: The (a) time taken and (b) accuracy for computing euclidean distance on sketches of single entities from Dataset 2. It is seen that, higher the sketch vector size(k), higher the query latency but higher the accuracy.

Figure 3, we show the behavior of query time and accuracy of query evaluation using sketches for different entity counts. It can be seen that higher number of sketches incur higher retrieval and processing costs. However, higher number of sketches also also means higher accuracy or lesser error. Higher randomization introduced by more number of sketches reduces the error [10]. This exhibits the tradeoff between response time and accuracy accuracy mentioned earlier.

5.4. Entity aggregation

When aggregation is performed across multiple entities, using either histograms or sketches, the query-time behavior is quite as expected – the time taken increases linearly with the number of entities. However, while using sketches, we get the advantage of higher accuracy with higher number of entities, attributed to higher randomization. This is shown in Figure 4(b).

5.5. Range Tree

We have seen that while querying on linear representations such as histograms and sketches, the query time increases linearly with the query size. However, range trees give us the advantage of directly fetching histograms that have already been aggregated temporally. Figure 2(b,c) show the advantage of using range tree for queries with high query sizes.

We use the range tree structure in a flexible manner; it can contain a mix of sketches and histograms. Figure 4(a) shows the query times for range trees that are based purely out of histograms, purely out of sketches and the third one, a mix of both. The criteria for selecting a sketch or a histogram as a range tree node was based on size. If the sketch happens to be smaller, it becomes the value of the node, else it is the histogram. We can see that the hybrid one performs somewhat better than either of the pure configurations.

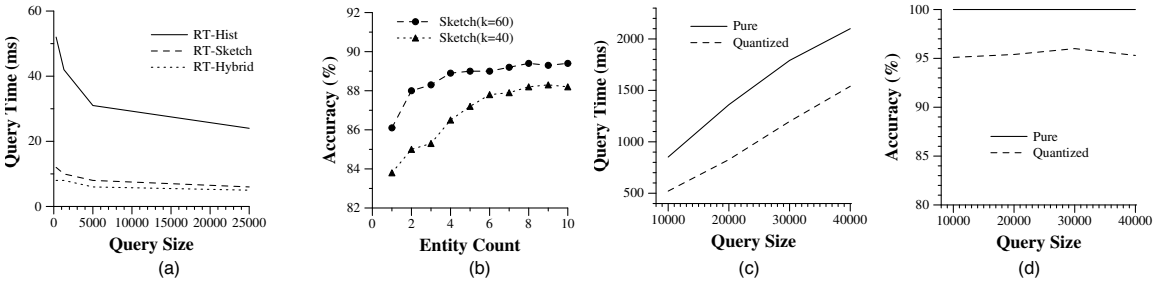

Figure 4: (a) Query times for different range tree configurations - purely histogram based, purely sketch based and a hybrid of the two, respectively. The entity count is 1 and performed on dataset 1. The hybrid one performs somewhat better than either of the pure configurations. (b) Accuracy increases with the number of entities aggregated in a query. We used sketches with size, $k = 40$ and $k = 60$, over 6 queries of size 400ms each on Dataset 2. (c) Query times and (d) Accuracy for histograms and the quantized version of histograms for different query sizes: the loss of about 5% accuracy, yields a speedup of around 25%.

5.6. Quantization

The principal advantage of quantization is that by rounding coefficients, we can reduce the space and achieve speedups. Quantization comes into effect in two forms - (a) storing lesser information and hence fetching less at query time, and, (b) being able to utilize memory in a more efficient manner by fitting the quantized representations in memory (reflected in Section 5.2). Figure 4(c,d) show the comparison of query times and accuracy of quantized histograms with non quantized ones in case of histograms as described in Section 4.3. The disproportionately higher speedup compared to the small loss in accuracy is attributed to the significant reduction in storage size due to quantization, leading to faster fetching times; whereas, the final answer is only effected by a fraction.

6. Discussion

In this paper, we presented an approach to perform approximate multi-resolution temporal aggregation queries in timeseries efficiently. We came up with a novel combination of data sketching, wavelet-style differencing for temporal compression, and geometric quantization to create compact representations of large timeseries. We showed, theoretically and empirically, how to track error with such different representations. We also presented FAQ (Fast Approximate Query-able), a data structure spanning disk and memory, that uses multiple representations of timeseries data to facilitate a trade-off choice between accuracy and query time. Constrained to satisfy a specified error guarantee in the query, FAQ's objective is to evaluate the the query in the least response time. We show a 92-fold speed up for euclidean distance queries using FAQ over a raw histogram representation. Since timeseries data is omnipresent today and given the evaluation of temporal aggregates in applications such as trend analysis, pattern discovery, drift detection, anomaly detection etc., we believe such a structure will be useful for speeding up many offline and online tasks in stream and historical data management. This work has given rise to a number of future directions of

work. Firstly, we are engaged in deploying FAQ for use in multiple real world application systems.Secondly, we are exploring the use of such representations in other aggregates also in the context of different types of queries, such as, sliding window queries over streaming data. Finally, we are also working on a data-driven optimization of the FAQ structure by auto tuning of parameters, such as tree arity and sketch sizes.

References

[1] Noga Alon, Yossi Matias, and Mario Szegedy. The space complexity of approximating the frequency moments. In *ACM Symposium on Theory of Computing*, pages 20–29, 1996.

[2] Cláudia M Antunes and Arlindo L Oliveira. Temporal data mining: An overview. In *KDD Workshop on Temporal Data Mining*, pages 1–13, 2001.

[3] Yoram Bachrach, Ralf Herbrich, and Ely Porat. Sketching algorithms for approximating rank correlations in collaborative filtering systems. In *International Symposium on String Processing and Information Retrieval*, pages 344–352, 2009.

[4] Ziv Bar-Yossef, T. S. Jayram, Ravi Kumar, D. Sivakumar, and Luca Trevisan. Counting distinct elements in a data stream. In *International Workshop on Randomization and Approximation Techniques*, pages 1–10, 2002.

[5] Andrei Z. Broder, Moses Charikar, Alan M. Frieze, and Michael Mitzenmacher. Min-wise independent permutations (extended abstract). In *ACM STOC*, pages 327–336, 1998.

[6] Alessandro Camerra, Jin Shieh, Themis Palpanas, Thanawin Rakthanmanon, and Eamonn Keogh. Beyond one billion time series: indexing and mining very large time series collections with iSAX2+. *Knowledge and Information Systems*, pages 1–29, 2013.

[7] F.K.P. Chan, A.W.C. Fu, and C. Yu. Haar wavelets for efficient similarity search of time-series: with and without time warping. *IEEE TKDE*, 15(3):686–705, 2003.

[8] Moses Charikar, Kevin Chen, and Martin Farach-Colton. Finding frequent items in data streams. In *Proceedings of the 29th International Colloquium on Automata, Languages and Programming*, pages 693–703, 2002.

[9] Moses S. Charikar. Similarity estimation techniques from rounding algorithms. In *ACM STOC*, pages 380–388, 2002.

[10] Graham Cormode, Minos Garofalakis, Peter J. Haas, and Chris Jermaine. Synopses for massive data: Samples, histograms, wavelets, sketches. *Found. Trends databases*, 4:1–294, January 2012.

[11] Graham Cormode and S. Muthukrishnan. An improved data stream summary: The count-min sketch and its applications. *J. Algorithms*, 55(1):58–75, April 2005.

[12] Philippe Flajolet and G. Nigel Martin. Probabilistic counting algorithms for data base applications. *J. Comput. Syst. Sci.*, 31(2):182–209, September 1985.

[13] Sorabh Gandhi, Luca Foschini, and Subhash Suri. Space-efficient online approximation of time series data: Streams, amnesia, and out-of-order. In *IEEE ICDE*, pages 924–935, 2010.

[14] Deke Guo, Jie Wu, Honghui Chen, Ye Yuan, and Xueshan Luo. The dynamic bloom filters. *IEEE TKDE*, 22(1):120–133, Jan 2010.

[15] Piotr Indyk. Stable distributions, pseudorandom generators, embeddings, and data stream computation. *J. ACM*, 53(3):307–323, May 2006.

[16] Udayan Khurana and Amol Deshpande. Efficient snapshot retrieval over historical graph data. In *IEEE ICDE*, pages 997–1008, 2013.

[17] Hyun Duk Kim, Danila Nikitin, ChengXiang Zhai, Malu Castellanos, and Meichun Hsu. Information retrieval with time series query. In *Proceedings of the 2013 Conference on the Theory of Information Retrieval*, page 14, 2013.

[18] Srivatsan Laxman and P Shanti Sastry. A survey of temporal data mining. *Sadhana*, 31(2):173–198, 2006.

[19] Ping Li. Estimators and tail bounds for dimension reduction in l α $(0 < \alpha \leq 2)$ using stable random projections. In *ACM-SIAM SODA*, pages 10–19, 2008.

[20] Ping Li and Cun-Hui Zhang. Entropy estimations using correlated symmetric stable random projections. In *NIPS*, pages 3185–3193, 2012.

[21] Theophano Mitsa. *Temporal data mining*. CRC Press, 2010.

[22] Abdullah Mueen, Suman Nath, and Jie Liu. Fast approximate correlation for massive time-series data. In *ACM SIGMOD*, pages 171–182, 2010.

[23] Vit Niennattrakul, Pongsakorn Ruengronghirunya, and Chotirat Ann Ratanamahatana. Exact indexing for massive time series databases under time warping distance. *Data Mining and Knowledge Discovery*, 21(3):509–541, 2010.

[24] Andrew R Post and James H Harrison Jr. Temporal data mining. *Clinics in Laboratory Medicine*, 28(1):83–100, 2008.

[25] Thanawin Rakthanmanon, Bilson J. L. Campana, Abdullah Mueen, Gustavo E. A. P. A. Batista, M. Brandon Westover, Qiang Zhu, Jesin Zakaria, and Eamonn J. Keogh. Searching and mining trillions of time series subsequences under dynamic time warping. In *ACM SIGKDD*, pages 262–270, 2012.

[26] Galen Reeves, Jie Liu, Suman Nath, and Feng Zhao. Managing massive time series streams with multi-scale compressed trickles. In *VLDB Endowment*, volume 2.1, pages 97–108, 2009.

[27] Jin Shieh and Eamonn Keogh. iSAX: disk-aware mining and indexing of massive time series datasets. *Data Mining and Knowledge Discovery*, 19(1):24–57, 2009.

[28] Xiaoyue Wang, Abdullah Mueen, Hui Ding, Goce Trajcevski, Peter Scheuermann, and Eamonn Keogh. Experimental comparison of representation methods and distance measures for time series data. *Data Mining and Knowledge Discovery*, 26(2):275–309, 2013.