Noname manuscript No.
(will be inserted by the editor)

Continuous Decaying of Telco Big Data with Data Postdiction

Constantinos Costa · Andreas Konstantinidis · Andreas Charalampous · Demetrios Zeinalipour-Yazti · Mohamed F. Mokbel

Abstract In this paper, we present two novel decaying operators for Telco Big Data (TBD), coined *TBD-DP* and *CTBD-DP* that are founded on the notion of *Data Postdiction*. Unlike data prediction, which aims to make a statement about the future value of some tuple, our formulated *data postdiction* term, aims to make a statement about the past value of some tuple, which does not exist anymore as it had to be deleted to free up disk space. *TBD-DP* relies on existing Machine Learning (ML) algorithms to abstract TBD into compact models that can be stored and queried when necessary. Our proposed *TBD-DP* operator has the following two conceptual phases: (i) in an offline phase, it utilizes a LSTM-based hierarchical ML algorithm to learn a tree of models (coined *TBD-DP* tree) over time and space; (ii) in an online phase, it uses the *TBD-DP* tree to recover data within a certain accuracy. Additionally, we provide three focused decaying methods that can be plugged into the operators we propose, namely: (i) FIFO-amnesia, which is based on the time that the tuple was created; (ii) SPATIAL-amnesia, which is based on the cellular tower's location related with the tuple; and (iii) UNIFORM-amnesia, which picks randomly the tuples to be decayed. Similarly, *CTBD-DP* enables the decaying of streaming data utilizing the *TBD-DP* tree to extend and update the stored models. In our experimental setup, we measure the efficiency of the proposed operator using a ∼10GB anonymized real telco network trace. Our experimental results in Tensorflow over HDFS are extremely encouraging as they show that *TBD-DP* saves an order of magnitude storage space while maintaining a high accuracy on the recovered data. Our experiments also show that *CTBD-DP* improves the accuracy over streaming data.

Constantinos Costa
University of Pittsburgh, Pittsburgh, PA 15213, USA
E-mail: costa.c@cs.pitt.edu;

Andreas Konstantinidis
Frederick University and University of Cyprus, 1678 Nicosia, Cyprus
E-mail: akonstan@cs.ucy.ac.cy;

Andreas Charalampous, Demetrios Zeinalipour-Yazti*
University of Cyprus, 1678 Nicosia, Cyprus
E-mail: {achara28, dzeina}@cs.ucy.ac.cy;

Mohamed F. Mokbel
Qatar Computing Research Institute
HBKU, Qatar and University of Minnesota, Minneapolis, MN 55455, USA
E-mail: mmokbel@hbku.edu.qa

Keywords telco big data; data decaying; data reduction; machine learning; spatio-temporal analytics;

1 Introduction

In recent years there has been considerable interest from *telecommunication companies (telcos)* to extract concealed value from their network data. Consider for example a telco in the city of Shenzhen, China, which serves 10 million users. Such a telco is shown to produce 5TB per day [39] (i.e., thousands to millions of records every second). Huang et al. [21] break their 2.26TB per day *Telco Big Data (TBD)* down as follows: (i) *Business Supporting Systems (BSS)* data, which is generated by the internal work-flows of a telco (e.g., billing, support), accounting to a moderate of 24GB per day and; (ii) *Operation Supporting Systems (OSS)* data, which is generated by the Radio and Core equipment of a telco, accounting to 2.2TB per day and occupying over 97% of the total volume.

Effectively storing and processing TBD workflows can unlock a wide spectrum of challenges, ranging from churn prediction of subscribers [21], city localization [40], 5G network optimization / user-experience assessment [22, 14, 29] and road traffic mapping [15]. Even though the acquisition of TBD is instrumental in the success of the above scenarios, Telcos are reaching a point where the data they collect is more than what they could possibly exploit. This has the following two implications: (i) it introduces a significant financial burden on the operator to store the collected data locally. Notice that the deep storage of data in public clouds, where economies-of-scale are available (e.g., AWS Glacier), is not an option due to privacy reasons; and (ii) it imposes a high computational cost for accessing and processing the collected data. For example, a petabyte Hadoop cluster, using between 125 and 250 nodes, costs $\sim$1M USD [30] and a linear scan of 1PB would require almost 15 hours. Additionally, in [26] it is shown that the amount of storage doubles every year and storage media costs decline only at a rate of less than 1/5 per year. Finally, high-availability storage mandates low-level data replication (e.g., in HDFS the default data replication is 3).

Consequently, we claim that the vision of infinitely storing all IoT-generated velocity data on fast high-availability or even deep storage will gradually become too costly and impractical for many analytic-oriented processing scenarios.

To this end, *data decaying* [24, 23] (or data rotting) has recently been suggested as a powerful concept to complement traditional data reduction techniques [12, 4], e.g., sampling, aggregation (OLAP), dimensionality reduction (SVD, DFT), synopsis (sketches) and compression. Data decaying refers to *"the progressive loss of detail in information as data ages with time"*. In data decaying recent data retains complete resolution, which is practical for operational scenarios that can continue to operate at full data resolution, while older data is either compacted or discarded [24, 23, 14]. Additionally, the decaying cost can be amortized over time, matching current trends in micro-batching (e.g., Apache Spark). Unfortunately, data decaying currently relies on rather straightforward methodologies, such as rotational decaying (i.e., FIFO) [24], or decaying based on specific queries [14] rather than the complete dataset itself. Our aim in this work is to expand upon these developments to provide more intelligent and generalized decaying operators.

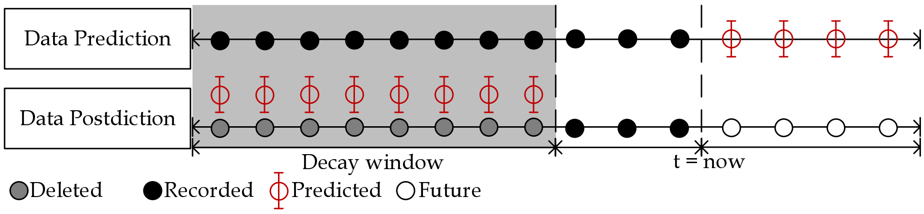

Fig. 1 Data Prediction (top): aims to find the future value of some tuple. **Data Postdiction (bottom)**: aims to recover the past value of some tuple, which has been deleted to reduce the storage requirements, using a ML model.

In this paper, we revisit our novel decaying operator for Telco Big Data, coined *TBD-DP (Data Postdiction)* [16] (see Figure 1) and present a new data decaying operator that can cope with streaming data, coined *CTBD-DP*. Unlike data prediction, which aims to make a statement about the future value of some tuple in a TBD store, data postdiction aims to make a statement about the past value of some tuple that does not exist anymore, as it had to be deleted to free up space. *TBD-DP* relies on existing Machine Learning (ML) algorithms to abstract TBD into compact models that can be stored and queried when necessary. Our proposed *TBD-DP* operator has the following two conceptual phases: (i) in an offline phase, it utilizes a LSTM-based hierarchical ML algorithm to learn a tree of models (coined *TBD-DP* tree) over time and space; (ii) in an online phase, it uses the *TBD-DP* tree to recover data with a certain accuracy. Additionally, *CTBD-DP* consumes newly generated data streams that need to be decayed in batch mode by updating the existing *TBD-DP* tree, on the fly. Particularly, *CTBD-DP* retrieves all the stored models based on the records in a batch and updates the models through new ML iterations.

We claim that the LSTM model is capturing the essence of the past through its short and long-term dependencies, similarly to how the brain retains both recent information and important old information at a high resolution.

To understand the operational aspects of our proposed operators, consider Figure 2, where we show how incoming telco data signals are absorbed by the TBD architecture and stored on high-availability and fast storage (i.e., D). This helps to carry out operational tasks (e.g., alerting services and visual analytics) with full data resolution. Subsequently, in the first phase of *TBD-DP*, we utilize a specialized *Recurrent Neural Network (RNN)* composed of *Long Short Term Memory (LSTM)* units, which has the ability to detect long-term correlations in activity data and the trained model has a small disk space footprint [25]. This enables *TBD-DP* to utilize minimum storage capacity of the decayed data by representing them with LSTM models on the disk media (D') and provide real-time postdictions with high accuracy in a subsequent recovery phase, which will be initiated on-demand (i.e., whenever some high-level operator requests the given data blocks).

This paper builds on our previous work in [16], in which we presented the preliminary design and results of our *TBD-DP* operator. In this paper we propose several new improvements, particularly a continuous data postdiction operator, coined *CTBD-DP*, as well as several pluggable decaying focus functions. All our propositions are evaluated using real telco data in a prototype architecture we have developed. The overall contributions of our work are summarized as follows:

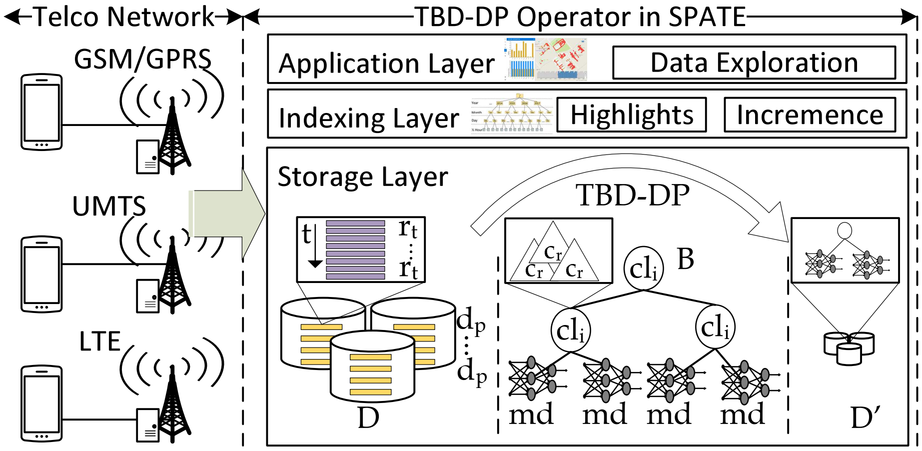

Fig. 2 System Model: The TBD-DP operator works on the storage layer of a typical TBD stack and abstracts the incoming data signals (D) into abstract models (md) that are organized in a tree data structure (B).

- We present a TBD decay operator that deploys the notion of data postdiction using off-the-shelf LSTM-based prediction models.
- We propose the DP-tree, which is a hierarchical index to organize the generated models in a data structure to enable the efficient recovery of data when necessary.
- We propose *CTBD-DP*, which is a continuous decay operator that utilizes data postdiction in order to process streaming data.
- We propose the design and implementation of multiple decaying functions, namely FIFO-amnesia, which is based on the timestamp that the tuple was created; SPATIAL-amnesia, which is based on the cellular tower's location and UNIFORM-amnesia, which picks randomly the tuples that will be decayed.
- We measure the efficiency of the proposed operator using a ∼10GB anonymized telco network trace, showing that our operators can be a premise for efficient TBD analytics in the future. We also summarize a prototype architecture and user interface we have developed for the management of TBD.

The remainder of the paper is organized as follows: In Section 2, we classify the related work into three categories. Section 3 formalizes our system model, assumptions and problem definition. In Section 4, we introduce the proposed *TBD-DP* operator and discuss its two internal algorithms. In Section 5, we present the proposed *CTBD-DP* operator along with *Continuous Construction* algorithm. Section 6 presents a complete prototype architecture that integrates our operators. Section 7 presents our experimental methodology and the results of our evaluation and Section 8 concludes the paper.

2 Related Work

This section provides a concise coverage of related work in Telco Big Data, which appears more extensively as an advanced seminar in [13]. It also briefly touches upon issues of data reduction that are necessary to put into perspective the contributions of this work.

2.1 Telco Big Data (TBD) Research

Telco research can be roughly classified into the following three categories: (i) real-time analytics and detection; (ii) predicting user behavior; and (iii) privacy. There is also Telco research that focus on applications that Telcos can use to improve their services and revenue. Such kind of literature, however, is orthogonal to the topic of this article.

Real-time Analytics and Detection: Zhang et al. [39] have developed *OceanRT* for managing large spatiotemporal data, such as Telco OSS data, running on top of cloud infrastructure. It contains a novel storage scheme that optimizes queries with joins and multi-dimensional selections. Yuan et al. [37] present *OceanST* that features: (i) an efficient loading mechanism of ever-growing Telco MBB data; and (ii) new spatiotemporal index structures to process exact and approximate spatiotemporal aggregate queries in order to cope with the huge volume of MBB data. Iyer et al. [22] present *CellIQ* to optimize queries such as "spatiotemporal traffic hotspots" and "handoff sequences with performance problems". It represents the snapshots of cellular network data as graphs and leverages on the spatial and temporal locality of cellular network data.

Braun et al. [9] developed a scalable distributed system that efficiently processes mixed workloads to answer event stream and analytic queries over Telco data. Bouillet et al. [8] proposed a system on top of IBM's InfoSphere Streams middleware that analyzes 6 billion CDRs per day in real-time. Abbasoğlu et al. [1] present a system for maintaining call profiles of customers in a streaming setting by applying distributed stream processing.

Experience, Behavior and Retention Analytics: Huang et al. [21] empirically demonstrate that customer churn prediction performance can be significantly improved with telco big data. Although BSS data have been utilized in churn prediction very well in the past decade, the authors show how with a primitive Random Forest classifier telco big data can improve churn prediction accuracy from 68% to 95%. Luo et al. [29] propose a framework to predict user behavior involving more than one million telco users. They represent users as documents containing a collection of changing spatiotemporal "words" that express user behavior. By extracting the users' space-time access records from MBB data, they learn user-specific compact topic features that they use for user activity level prediction.

Privacy: Hu et al. [20] study Differential Privacy for data mining applications over telco big data and show that for real-word industrial data mining systems the strong privacy guarantees given by differential privacy are traded with a 15% to 30% loss of accuracy. Privacy and confidentiality are critical for telcos' reliability due to the highly sensitive attributes of user data located in CDR, such as billing records, calling numbers, call duration, data sessions, and trajectory information.

Table 1 Summary of notations

Notation	Description
p, d_p, D	Ingestion period, data snapshot of one p, set of all d_ps
t, r_t	Timestamp within an ingestion cycle, record at t
C, c_r, cl_i	Set of all cell towers, Cell of record r, cluster of records $i = 1, \ldots, k$
md_i, MD	LSTM model of cluster cl_i, set of all models
f	Decaying factor: percentage of data to be removed
df	Decaying focus: ordering algorithm that the decay function will follow
b	A batch of the data snapshots from the telco network

2.2 Compressing Incremental Archives

Domain-specific compression techniques are often adopted for compressing spatiotemporal climate data [7], text document collections [35], scientific simulation floating point data [28, 31, 33, 5], and floating point data streams [10]. Moreover, several research studies [18, 36, 6] have utilized differential compression techniques for studying the trade-off between compression ratio and decompression times for incremental archival data. None of these prior research works, however, has been proposed for dealing with data decaying in Telco-specific distributed systems.

2.3 Data Synopsis

Sampling refers to the process of randomly selecting a subset of data elements from a relatively large dataset. Sophisticated techniques, such as Bernoulli and Poisson sampling, choose data elements using probabilities and statistics. Chaudhuri et al. [11] proposed *stratified sampling* where the probability of the selection is biased. In order to encounter the big data sampling issue, Zeng et al. [38] implemented G-OLA, which is a model that generalizes online aggregation in order to support general OLAP queries utilizing delta maintenance algorithms. Particularly, BlinkDB [3] allows users to choose the error bounds and the response time of query using dynamic sampling algorithms. SciBORQ [32] is a framework that allows the user to choose the quality of the query result based on multiple interesting data samples called impressions.

Several works have adapted the sampling processes to create synopsis of data in order to achieve low response time for ad-hoc queries [32]. Data sketches [12] are compact data structures that enable to efficiently estimate the count of occurences in massive data (contrary to Bloom filters, it encodes a potentially massive number of item types in a small array). Additionally, Wei et al. proposed persistent sketches that can answer queries at any prior time [34] and have the ability to merge in order to answer a generalization query [2].

3 System Model and Problem Formulation

This section formalizes our system model, assumptions and problem. The main symbols and their respective definitions are summarized in Table 1.

A typical Telco system, illustrated in Figure 2, is composed of the Telco network, which is responsible for providing telecommunication services, and a Telco data management system, such as SPATE [14], which is responsible for the efficient analytical exploration of Telco datasets. The data arrives at the data center in batches, called henceforth data snapshots noted by d_p, in the form of horizontally segmented files within an ingestion period p. A snapshot d_p contains multiple records r_t created at a certain timestamp t. Each record r_t consists of a predefined set of attributes including the cell id c_r that represents the spatial information inherent within the Telco network. Particularly, each cell id c_r corresponds to a cell that covers a geographical cellular area that usually spans hundreds of meters or even kilometers. Finally, the cells are spatially grouped into clusters $cl_i, i = i \dots k$ for facilitating the postdiction process by creating a model $md_i, i = i \dots k$ for each cl_i as this will be explained in the next section.

3.1 Problem Formulation

Research Goal. *Given a Telco setting, this work aims at achieving a pre-specified decaying of TBD with minimum additional storage space capacity and being able to recover the decayed data accurately and efficiently.*

The efficiency of the proposed techniques to achieve the above goal is measured by the following objectives:

Definition 3.1: **Storage Capacity (S)** is the total storage space required for achieving decaying of data based on a pre-specified decaying factor f.

Definition 3.2: **Accuracy (*NRMSE*)** is the percentage of the correctly recovered decayed data. It is measured by the normalized root-mean-square error, which is the normalized difference between the actual data ($x_{1,t}$) and the predicted data ($x_{2,t}$), where t is a discrete time point and y_{max}, y_{min} the maximum and minimum observed differences, formally:

$$NRMSE = \frac{\sqrt{\frac{1}{n}\sum_{t=1}^{n}(x_{1,t} - x_{2,t})^2}}{(y_{max} - y_{min})}$$

4 The *TBD-DP* operator

In this section, we introduce the *TBD-DP* operator and discuss its two internal algorithms, namely, the *Construction* (data model creation) and the *Recovery* (data recreation), which capture its core functionality as illustrated in Figure 3.

The *Construction* algorithm can be triggered either by the user, or automatically when the total storage capacity reaches a certain level. In both cases, the data are initially clustered based on spatial characteristics and then ordered based on temporal information. Finally, postdiction models based on the LSTM machine learning approach are generated for each cluster and the real data is decayed by $f\%$. The *Recovery* algorithm utilizes the postdiction models for retrieving the decayed data by adopting a proposed DP-tree based algorithm.

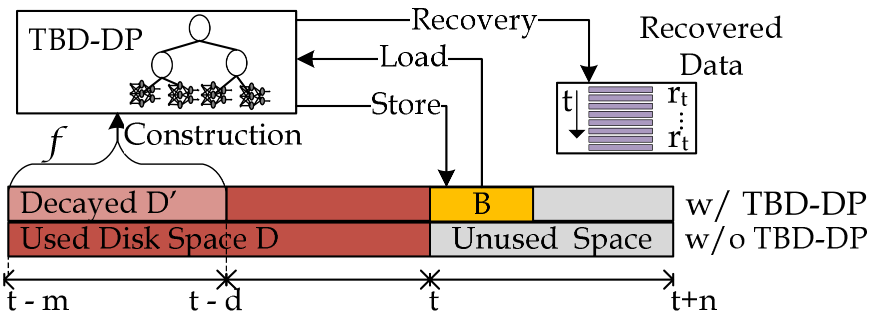

Fig. 3 *TBD-DP* Operator Overview.

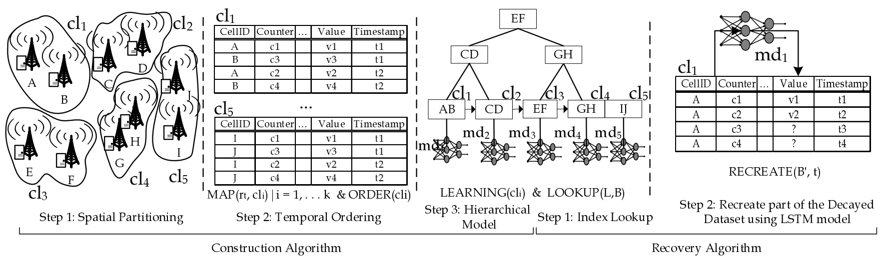

Fig. 4 The conceptual steps of the proposed *TBD-DP* construction and recovery algorithm.

4.1 Construction Algorithm

Algorithm 1 outlines the major steps of the construction algorithm. Initially, the decaying factor f specifies the percentage of the whole dataset D that will be decayed, and consequently the decayed subset $D' \subseteq D$ that will be utilized for generating the postdiction models. In the spatial partitioning step (Step 1 - lines 11-14), $k \leq |C|$ clusters are created by using the cell tower locations. Particularly, each cluster $cl_i, i = 1, \ldots, k$ is represented by a cell tower (in cases where $k < |C|$ then the closest cell towers are merged using a kNN approach until we finally generate k clusters). The clustering step has a two-fold contribution for the *CTBD-PD* operator: (i) it takes advantage of the spatio-temporal circularity of the telco data at each cellular tower in order for the machine learning approach to create a more biased and therefore more accurate models for each single cellular tower; the circularity of the data is evident from our data analysis, since there is a similar pattern that is repeated every some time for every cellular tower location. (ii) the clustering step will also reduce the time needed for retrieving the decayed data at each single query, since the time needed for locating the correct model and retrieve a number of decayed data associated to one (or a group) cellular tower is much less than "postdicting" the whole dataset. Then the *MAP* function associates all records $r_t \in D'$ with the previously created clusters by taking into consideration their cell id c_r attribute. By the end of this function execution, k clusters of cell towers with their associate records will be created. Then all records of each cluster are ordered based on their timestamp or their cell tower's location or uniformly (i.e., time originally generated) by using the *ORDER* function of the temporal ordering step (Step 2 - lines 15-17). This allows the neural network to be created correctly based on a continuous time series using the FIFO-amnesia decay function as described in Section 4. Finally, the learning step (Step 3 - lines 18-21) generates k postdiction models md_i for each cluster cl_i by using a specialized Recurrent Neural Network (RNN) known as Long Short Term Memory (LSTM) model [19].

Specifically, the *LEARNING* function generates, for each cluster at each iteration, an LSTM model that relies on a structure called a memory cell, which is composed of four main elements: an input gate, a neuron with a self-recurrent connection (a connection to itself), a forget gate and an output gate. A memory cell is updated at every time-step by using the following parameters and equations:

- x_t is the input to the memory cell layer at time-step t
- W_i, W_f, W_C and W_o are weight matrices
- b_i, b_f, b_C and b_o are bias vectors

The forget gate layer:

$$f_t = \sigma(W_f \cdot [h_{t-1}, x_t] + b_f),$$

decides what information are going to be thrown away from the memory cells. The input gate layer:

$$i_t = s(W_i \cdot [h_{t-1}, x_t] + b_i),$$

decides which values to be updated. The tanh layer decides what new information we are going to store in the memory cells using:

$$\widetilde{C_t} = tanh(W_C \cdot [h_{t-1}, x_t] + b_C).$$

Moreover, the update memory cells function:

$$C_t = f_t \times C_{t-1} + i_t \times \widetilde{C_t},$$

used to forget the things decided to be forgotten earlier and scale the new candidate values by a pre-specified state value.

Finally, the update hidden cells function:

$$o_t = \sigma(W_o \cdot [h_{t-1}, x_t] + b_o)$$

and a sigmoid layer that decide what parts of the cell state to output,

$$h_t = o_t * tanh(C_t).$$

The *Construction* algorithm outputs a set of postdiction models B in a DP-tree for facilitating the recovery algorithm that follows. At the end of the *Construction* algorithm execution, the D' set of data is removed for saving storage space and it is conceptually replaced by the final B set of postdiction models, where $|B| << |D'|$.

Decay Principle of TBD-DP: *Decaying* refers to the progressive loss of detail in information as data ages with time until it has completely disappeared. Kersten refers to the existence of data fungus in [23] with a decaying operator coined *"Evict Grouped Individuals (EGI)"*. The given EGI operator performs biased random decaying, resembling the rotting process in nature (e.g., in fruits with fungus). In our previous work [14], we used the *First-In-First-Out (FIFO)* data fungus, i.e., *"Evict*

Algorithm 1 - *TBD-DP* Construction Algorithm

Input: Dataset D, C set of cell towers, Number of clusters k, df decaying focus
Output: B: Set of models MD (DP-tree structure)

procedure ORDER(cl_i,df)
  switch df **do**
    case $FIFO$ ▷ Sort records in clusters t based on timestamp.
      return $SORT_{FIFO}(cl_i)$
    case $SPATIAL$ ▷ Sort records based on c_r cell tower's location.
      return $SORT_{SPATIAL}(cl_i)$
    case $UNIFORM$ ▷ Sort records based on a uniform distribution.
      return $SORT_{UNIFORM}(cl_i)$
end procedure
  ▷ **Step 0: Decaying Pre-processing**
$D' \leftarrow f$ of D ▷ Select $f\%$ of D to be decayed
  ▷ **Step 1: Spatial Partitioning**
Create $k \leq |C|$ clusters cl_i ▷ Use cell towers locations
for all $r_t \in D'$ **do**
  $cl_i \leftarrow MAP(r_t, cl_i)|i = 1, \ldots k$ ▷ Associate records to clusters
end for
  ▷ **Step 2: Ordering**
for $i = 1$ to k **do**
  $cl_i \leftarrow ORDER(cl_i, df)$ ▷ Sort records in clusters
end for
  ▷ **Step 3: Hierarchical Model**
for $i = 1$ to k **do**
  $md_i \leftarrow LEARNING(cl_i)$ ▷ Create an LSTM model for each cl_i
  Insert md_i in B
end for

Oldest Individuals", which retains full resolution for recent data but abstracts older data into compact aggregation models. Both EGI and FIFO do not retain full resolution for important instances that occurred in the past. Consequently, data would have been rotted and purged either randomly or based on its timestamp. We call this the *long-term dependency* problem. In this work, we chose a radically new decaying technique that could be termed as *LSTM data fungus*, which is explicitly designed to avoid the *long-term dependency* problem. Particularly, the *TBD-DP* operator replaces the data with abstract LSTM models, which capture the essence of the past, i.e., both recent data and important old data is retained at the highest possible resolution.

FIFO-amnesia decays data based on the time that tuples were ingested into the system and it is the most natural decaying technique. This mimics the way that humans forget old activities. In our case, this means that the older tuples can be more easily forgotten from the system.

Example: Consider the scenario in Figure 4 in which there are 10 cell towers $\{A, ..., J\}$. First, the *Construction* algorithm creates $k = 5$ clusters $\{cl_1,...,cl_5\}$ denoted with the solid line that surrounds the cell towers in Step 1 of Figure 4 (left). The *MAP* function associates the records to a cluster based on the cell id c_r (e.g., all records related to A and B are grouped into cl_1). Then, the *ORDER* function sorts the records of each cluster based on their timestamp t as shown in Step 2 of Figure 4 (center). This will produce similar result of tuples to be decayed

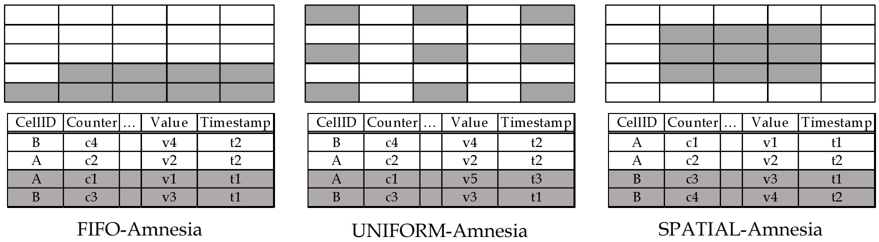

Fig. 5 Applying different decaying focuses *(df)* using the ORDER procedure

denoted with gray color in Figure 5 (left). Finally, for each cluster cl_i a model md_i is trained and inserted into a DP-tree index using the cell ids as keys, as shown in Figure 4 (right).

UNIFORM-amnesia decays data in a uniform random manner. During the decaying procedure each tuple has the same probability to be decayed.

Example: Consider the same scenario in which the *Construction* algorithm creates $k = 5$ clusters $\{cl_1,...,cl_5\}$ and associates the records with the records to the cluster based on the cell id c_r. The only difference is that the *ORDER* function is based on a different decaying focus. This will produce similar result of tuples to be decayed based on a uniform random distribution, denoted with gray color in Figure 5 (center).

SPATIAL-amnesia decays data based on the spatial attribute of each record (e.g., cell id). This reflects the decay process as a data fungus or mold that is spread on the nearby areas as described in [24].

Example: Consider the same scenario with the 10 cell towers $\{A, ..., J\}$ and the $k = 5$ clusters $\{cl_1,...,cl_5\}$. The only difference, in this case, is that the *ORDER* function is based on a SPATIAL-amnesia decaying focus. This will produce similar result of tuples to be decayed based on the cell tower location, denoted with gray color in Figure 5 (right).

4.2 Recovery Algorithm

Algorithm 2 outlines the *Recovery* algorithm that utilizes the DP-tree structure (B) of postdiction models of Algorithm 1 for retrieving a selected subset from the decayed data, i.e., $pD' \subseteq D'$. For doing this, the *Recovery* algorithm inputs the set of models B as well as some spatiotemporal information L and R that will specify the amount of the decayed data to be retrieved. For example, L can be a cellular tower's location or a user's location associated to a cellular tower and R can be a range of timestamps, within which a number of records were generated and stored in D'. In any case, L and R will be utilized by the DP-tree *LOOKUP* function for deciding a subset of models $B' \subseteq B$ in line 13 that will be used for creating the pD' dataset in line 15.

Example: Consider the scenario of Figure 4 (Recovery Algorithm) where the data of cell tower A (part of cl_1) needs to be recovered for timestamps t_1,...,t_4. *LOOKUP*

Algorithm 2 - *TBD-DP* Recovery Algorithm

Input: L: spatial input; R: temporal input; B: set of postdiction models in a DP-tree structure
Output: Partial decayed dataset pD'

procedure LOOKUP(k,*node*) ▷ The number of children is b.
    if *node* is a leaf **then**
        return *node*
    end if
    switch k **do**
        case $k < k_0$
            return LOOKUP(k, p_0)
        case $k_i \leq k < k_{i+1}$
            return LOOKUP(k, p_{i+1})
        case $k_d \leq k$ ▷ Each node has at most $d \leq b$
            return LOOKUP(k, p_{d+1})
end procedure
▷ **Step 1: Index Lookup**
$B' \leftarrow LOOKUP(L, B)$ ▷ Select a subset of postdiction models
▷ **Step 2: Recreate part of the Decayed Dataset using LSTM model**
for all $t \in R$ **do**
    $pD' = RECREATE(B', t)$ ▷ Retrieve decayed data of specific time periods.
end for

retrieves the LSTM model md_1 for cluster cl_1 created from all records related to cell towers A and B as shown in Step 1 of the given figure. In Step 2, the *Recovery* algorithm recreates the values of cell tower A for each timestamp t recovering in this way a part of the decayed data pD' using the selected LSTM model.

4.3 Performance Analysis

The secondary focus of *TBD-DP* is the efficient decaying of data and consequently the minimization of TBD storage space while maintaining a high accuracy during data recovery.

According to Definition 3.1 the total storage space S is equal to the actual data minus the decayed data based on f, plus any additional storage required by the decaying approach to achieve an optimal recreation of the decayed data. When there is no decaying $f = 0\%$ then $S = |D| + |B|$ (B could have been used for predicting future D values), which is the size of the actual (raw) data D and the size of the set of prediction models B. In the case of *TBD-DP*, $S = |D| - |D'| + |B|$, which is the actual data size minus the size of the decayed dataset $|D'| = |D| \times f\%$ plus the size of a set of models B, where $|D| >> |D'| + |B|$. When $f = 100\%$ then all data are decayed and the required storage space of *TBD-DP* is $S = |B|$. In the case of sampling, the storage space is equal to $S = |D| - |V|$, which is the actual data size minus a sample set $V = sampling(D', s)$ generated by sampling the decayed dataset D' with a pre-specified rate s. Note that $|D| - |D'| + |B| << |D| - |V|$ for a reasonable s that provides an *NRMSE* similar to *TBD-DP*.

According to Definition 3.2 the *NRMSE* measures the similarity of the decayed dataset D' and the recovered dataset pD'. Therefore, in cases where the decaying factor is $f = 0\%$, which corresponds to a low $|D'| = 0$ and no decaying is applied then $NRMSE = 0$ and when $f = 100\%$, which corresponds to a high $|D'| = |D|$ and

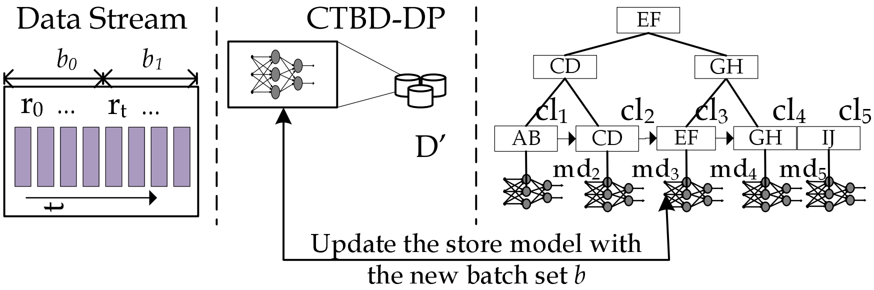

Fig. 6 *CTBD-DP* Operator Overview.

Algorithm 3 - *CTBD-DP* Continuous Construction Algorithm

Input: B, b, C set of cell towers, Number of clusters k, *df* decaying focus
Output: B: Set of updated models MD (DP-tree structure)

procedure ORDER(cl_i,*df*) ▷ Algorithm 1: (ORDER - lines 1-9)
end procedure
▷ **Step 0: Decaying Pre-processing**
$b' \leftarrow f$ of b ▷ Select $f\%$ of batch b to be decayed
▷ **Step 1: Spatial Partitioning**
Create $k \leq |C|$ clusters cl_i ▷ Use cell towers locations
for all $r_t \in b'$ **do**
  $cl_i \leftarrow MAP(r_t, cl_i)|i = 1, \ldots k$ ▷ Associate records to clusters
end for
▷ **Step 2: Ordering** ▷ Algorithm 1: (Step 2 - lines 15-17)
▷ **Step 3: Continuous learning & Hierarchical Model**
for $i = 1$ to k **do**
  $B' \leftarrow LOOKUP(L, B)$ ▷ Retrieve a subset of postdiction models
  $md_i \leftarrow LEARNING(B', cl_i)$ ▷ Create or Update an LSTM model for each cl_i in b'
  Insert/Update md_i in B
end for

all data are discarded then *NRMSE* $>> 0$. Moreover, it is reasonable to assume that in sampling, where a sample set V of the decayed data D' is permanently discarded with a sampling rate s then, its *NRMSE* (V, D') will be equal to the normalized difference between the sampled and the actual data. Finally, the *NRMSE* of the proposed *TBD-DP* will be equal to the normalized difference between the predicted data of the LSTM model and the actual data, i.e., *NRMSE* (pD', D').

5 The *CTBD-DP* operator

In this section, we introduce the proposed *CTBD-DP* operator and discuss the *Continuous Construction* (data model creation) algorithm, which captures its core functionality as illustrated in Figure 6. The *Recovery* algorithm remains the same with the *TBD-DP* operator.

Algorithm 3 outlines the major steps of the continuous construction algorithm. Initially, the decaying factor f specifies the percentage of the current batch b of the data stream that will be decayed, and consequently the decayed subset $b' \subseteq b$ that will be utilized for generating the postdiction models. In the spatial partitioning step (Step 1 - lines 4-7), $k \leq |C|$ clusters are created by using the cell tower locations. This allows us to construct or update less models based on the network topology resulting to less computations. Then the MAP function associates all records $r_t \in b'$ with the previously created clusters by taking into consideration their cell id c_r attribute. By the end of this function execution, k clusters of cell towers with their associate records will be created. Then all records of each cluster are ordered based on their timestamp or their cell tower's location or uniformly (i.e., time originally generated) by using the $ORDER$ function of the temporal ordering step (Step 2). This allows the neural network to be created correctly based on a continuous time series using the FIFO-amnesia decay function as described in Section 4. Finally, the learning step (Step 3 - lines 8-12) retrieves the previously k created models or generates k postdiction models md_i for each cluster cl_i by using the Long Short Term Memory (LSTM) model.

6 Prototype Description

We have developed a complete prototype architecture that integrates *TBD-DP* as part of the TBD Awareness project[1]. Our proposed architecture comprises of three layers (see Figure 2), namely Storage Layer, Indexing Layer and Application Layer.

The *Storage layer* passes newly arrived network snapshots through a lossless compression process storing the results on a replicated big data file system for availability and performance. This component is responsible for minimizing the required storage space with minimal overhead on the query response time. The intuition is to use compression techniques that yield high compression ratios but at the same time guarantee small decompression times. We particularly use GZIP compression that offers high compression/decompression speeds, with a high compression ratio and maximum compatibility with I/O stream libraries in a typical big data ecosystem we use. Additionally, this layer uses the *TBD-DP* operator in order to provide the decay methods for the next layer. The storage layer is basically only responsible for the leaf pages of the *SPATE* index described in the next layer.

The Indexing Layer uses a multi-resolution spatio-temporal index, which is incremented on the rightmost path with every new data snapshot that arrives (i.e., every 30 minutes). In addition, the component computes interesting event summaries, called "highlights", from data stored in children nodes and stores them at the parent node. For each data exploration query, the internal node that covers the temporal window of the query is accessed, and its highlights are used to answer the query.

The Application Layer implements the querying module and the *data exploration* interfaces, which receive the data exploration queries in visual or declarative mode and use the index to combine the needed highlights and snapshots to answer the query. *SPATE* is equipped with an easy-to-use map-based web interface layer that hides the complexity of the system through a simple and elegant web interface 7.

[1] TBD Awareness, https://tbd.cs.ucy.ac.cy/

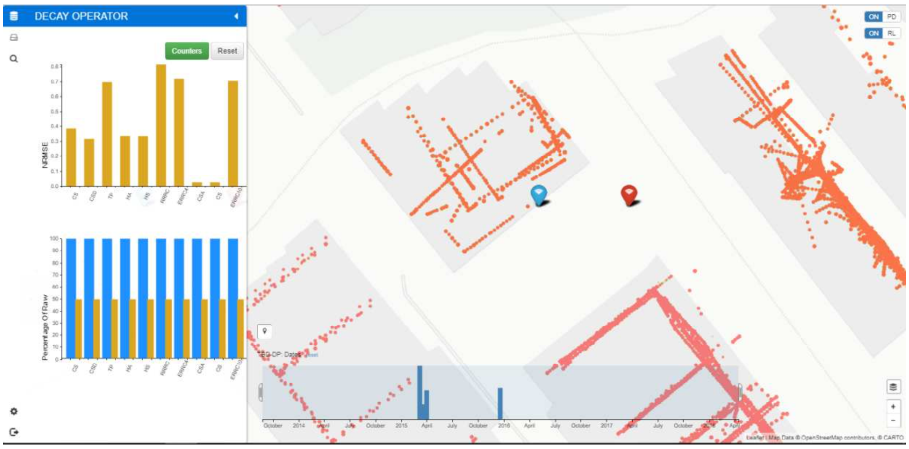

Fig. 7 The *TBD-DP* operator implemented inside the spatio-temporal SPATE architecture. The interface enables users to carry out high resolution visual analytics, without consuming enormous amounts of storage. The savings are quantified numerically with bar charts and visually with heatmaps.

7 Experimental Methodology and Evaluation

This section presents an experimental evaluation of our proposed operators. We start-out with the experimental methodology and setup, followed by two experiments. Particularly, in the first experiment, the performance of *TBD-DP* is compared against two baseline approaches and two decaying-based approaches with respect to various metrics on a set of anonymized datasets. The second experiment examines the influence of several control parameters on the performance of *TBD-DP*. The third experiment deals with the pluggable decaying focus methods while the fourth experiment deals with the evaluation of the *CTBD-DP* operator.

7.1 Methodology

This section provides details regarding the algorithms, metrics and datasets used for evaluating the performance of the proposed approach.

Testbed: Our evaluation is carried out on the DMSL VCenter IaaS datacenter, a private cloud, which encompasses 5 IBM System x3550 M3 and HP Proliant DL 360 G7 rackables featuring single socket (8 cores) or dual socket (16 cores) Intel(R) Xeon(R) CPU E5620 @ 2.40GHz, respectively. These hosts have collectively 300GB of main memory, 16TB of RAID-5 storage on an IBM 3512 and are interconnected through a Gigabit network. The datacenter is managed through a VMWare vCenter Server 5.1 that connects to the respective VMWare ESXi 5.0.0 hosts. Computing Nodes: The computing cluster, deployed over our VCenter IaaS, comprises of 4 Ubuntu 16.04 server images, each featuring 8GB of RAM with 2 virtual CPUs (@ 2.40GHz). The images utilize fast local 10K RPM RAID-5 LSI-Logic SCSI disks, formatted with VMFS 5.54 (1MB block size). Each node uses Hadoop v2.5.2.

We utilize anonymized measurements from a real Telco operator that comprises of 1192 real cell towers (i.e., 3660 cells of 2G, 3G and LTE networks) distributed in an area of 5,896 km^2. The cells are connected through a Gigabit network to a datacenter. Each cell tower keeps several UMTS/GSM network logs for the performance of the tower and forwards the information through the base station controller (BSC) or the radio network controller (RNC) to be stored. There is a CDR server that generates call detail records (CDRs) for incoming and outgoing calls in the enterprise. When a CDR is generated in the CDR server, the management server and third-party application can use SFTP to obtain the CDR from the CDR server. Then the Telco can query the CDRs for call/data information and check outgoing call/data fees with the carrier.

Algorithms: The proposed *TBD-DP* operator is compared with the following approaches:

- **RAW:** does not apply any decaying on the whole dataset.
- **COMPRESSION:** the decayed dataset is compressed with the *GZIP* library, which has been shown in [14] to offer the best balance between compression/decompression speeds, compression ratios and compatibility with I/O stream libraries.
- **SAMPLING:** a sampling method that picks every second item in the input stream, yielding a 50% sample size.
- **RANDOM:** uniformly randomly select one record from the decayed dataset.

Note that RAW and RANDOM are the baseline approaches used to demonstrate the trade-off between the storage capacity and the NRMSE objectives.

Datasets: We utilize an anonymized dataset of telco traces comprising of $\sim$ 100M network measurements records (NMS) and 3660 cells (CELL) coming from 2G, 3G and LTE antennas. The data traffic is created from about 300K objects and has a total size of $\sim$10GB. We constructed 6 realistic datasets from real TBD obtained through *SPATE* (depicted in Figure 8): described in Section 7.1 based on the *Key Performance Indicators (KPIs)* [27].

- **Calls (CS):** the number of calls ended normally during snapshot d_t.
- **Call Drops (CSD):** the number of calls dropped during snapshot d_t.
- **Handover Attempts (HA):** the amount of handovers into or from the cells attempted during a snapshot d_t.
- **Handovers (HS):** the number of successful handovers into or from the cells during a snapshot d_t.
- **Call Setup Attempts (CSA):** the amount of call setup processes attempted during snapshot d_t.
- **Call Setups (CE):** the amount of successful call setup processes during snapshot d_t.

The data distribution of the 6 realistic datasets, depicted in Figure 8, clearly shows that there is a repetitive pattern of values across the days of each KPI. Consequently, the *CTBD-DP* could be very efficient through the continuous learning in terms of accuracy.

Metrics: We evaluate the performance of *TBD-DP* using the metrics defined in Section 3.1 in all experiments:

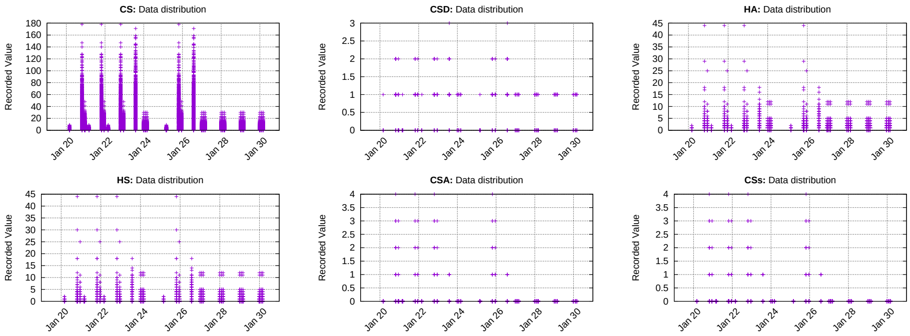

Fig. 8 Data distribution of an anonymized dataset of telco traces based on the counters.

- **Storage Capacity (S):** measures the total space that data and the DP-tree index occupy together, as a percentage of storage required by the RAW method (no decaying, no compression).
- **Normalized Root Mean Square Error (*NRMSE*):** measures the error of the recovered data D' using the NRMSE formula provided at the end of Section 3. A lower *NRMSE* value indicates a higher accuracy in the recovered data.

Parameters: In all experiments the simulation parameters were configured as follows: (i) decay factor $f = 50\%$ (indicating the percentage on which we execute the LSTM); (ii) the ML model is LSTM and the number of neurons 16 x 16. The influence of each of those parameters on the proposed approach is investigated individually in Experiment 2 by fixing the rest of the parameters accordingly.

7.2 Experiment 1: Performance Evaluation

In the first experiment, we evaluate the performance of the proposed *TBD-DP* operator against all four algorithms and over all datasets introduced in Section 7.1, with respect to space capacity (as a percentage to the RAW data) and accuracy (in terms of *NRMSE* on the decayed set of data).

Figure 9 clearly demonstrates the trade-off between the space capacity S and the *NRMSE* objectives on the results of the baseline approaches, since *RAW* (no decaying) approach obtained the worst possible $S = 100\%$ of the whole dataset, and the lowest error $NRMSE = 0$. In contrast, the *RANDOM* (almost all data are decayed) approach obtained the best possible $S = 50\%$ of the whole dataset and the worst $NRMSE \approx 100$ on the decayed dataset, for a decaying factor $f = 50\%$. The results of the three other approaches appear in between the results of the two baseline approaches. The proposed *TBD-DP* operator, however, provides around 25% and 50% better space capacity S compared to *COMPRESSION* and *SAMPLING* approaches, respectively. This is due to the fact that the additional space required by the set of LSTM models is much less than the sample set of *SAMPLING* and the compressed decayed dataset of *COMPRESSION*.

In terms of *NRMSE*, the *TBD-DP* outperforms the *SAMPLING* approach by 50%, on average, in all datasets. The *COMPRESSION* approach provides an opti-

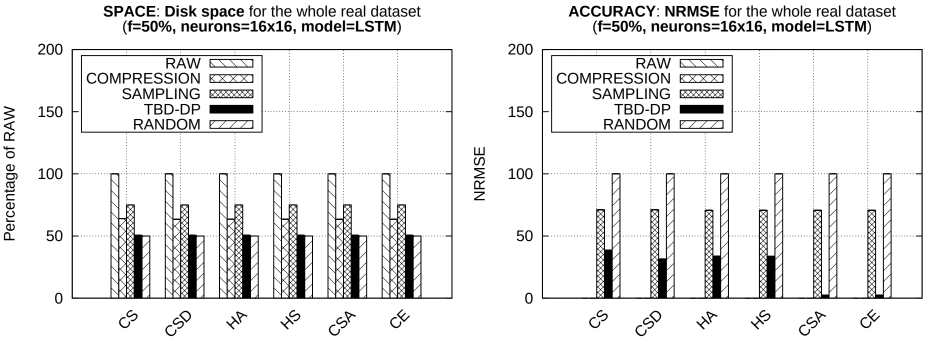

Fig. 9 Performance Evaluation: *TBD-DP* evaluation in terms of storage capacity S as a percentage to the RAW data (left) and accuracy in terms of *NRMSE* on the decayed set of data (right) in all datasets.

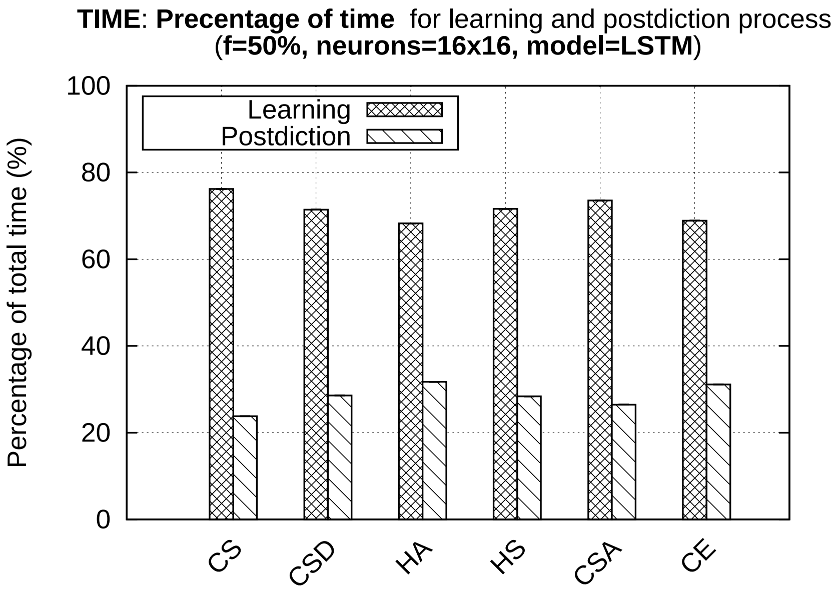

Fig. 10 Performance Evaluation: *TBD-DP* evaluation in terms of time percentage for the decayed set of data in all datasets.

mal $NRMSE = 0$, since it does not apply any prediction on the decayed data, but recovers them via decompression, when requested. The *COMPRESSION* approach however, can not be customized to achieve an even lower disk space occupancy. On the other hand, the *TBD-DP* can be configured, through its f parameter, to achieve a space occupancy that will fit the space budget of the application. This particular parameter will be investigated in the next experiment.

Figure 10 shows the total time for *TBD-DP* to complete the whole process including postdiction. The postdiction process takes much less time with respect to the learning process due to the LSTM network chain. The preprocessing step takes the majority of the required processing time due to network and disk IO.

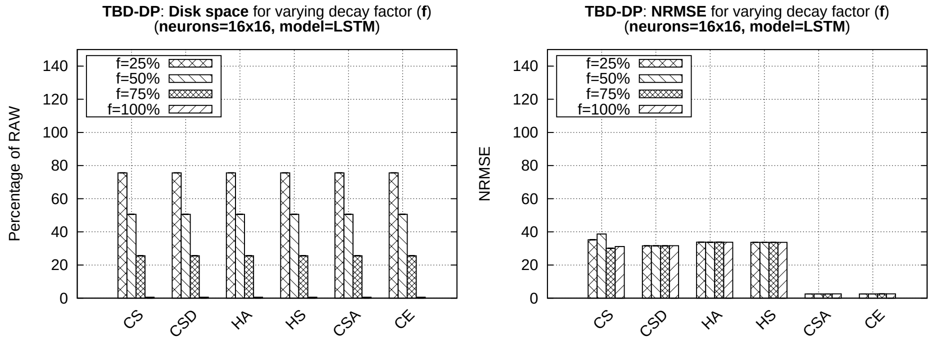

Fig. 11 Control Experiment - Decaying factor f: examining the storage capacity S and *NRMSE* of the proposed *TBD-DP* approach while varying f.

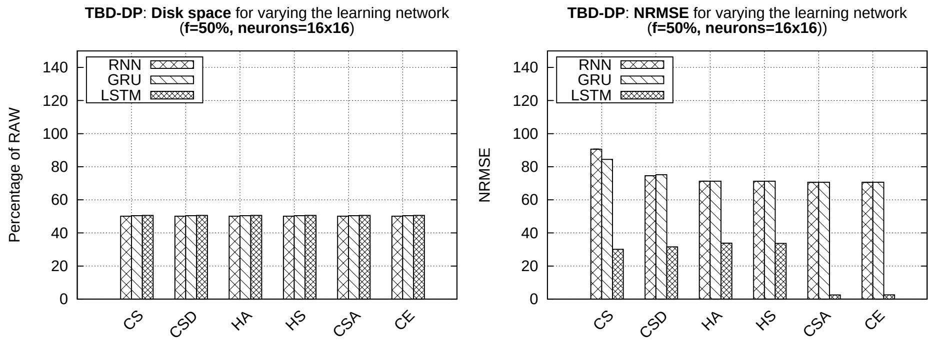

Fig. 12 Control Experiment - Learning Models: examining the storage capacity S and *NRMSE* of the proposed *TBD-DP* approach while combined with various ML models.

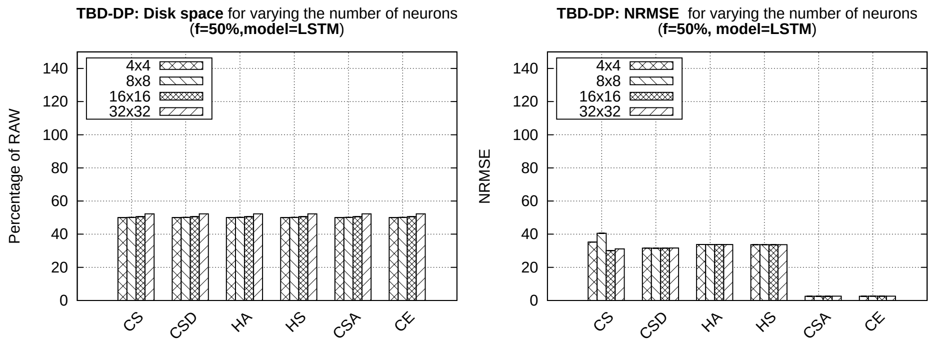

Fig. 13 Control Experiment - Number of neurons in LSTM: examining the storage capacity S and *NRMSE* of the proposed *TBD-DP* approach while varying the number of neurons in the LSTM.

7.3 Experiment 2: Control Experiments

In Experiment 2, we examine the influence of several control parameters on the performance of the proposed *TBD-DP* in terms of S and *NRMSE*. Specifically, we vary the decay factor (f), the ML models and the number of neurons on LSTM.

Figure 11 shows how the decaying factor f, and consequently the amount of data that will be decayed and represented by LSTM models, affect the S and *NRMSE* of the proposed *TBD-DP* operator. The results show that the storage capacity required by the *TBD-DP* decreases as the decaying factor increases, which is reasonable due to the fact that the highest f is, the more data need to be decayed and therefore more disk space will be released. The accuracy of the proposed *TBD-DP*, however, is not influenced, since *NRMSE* remains almost the same for all decaying factors, in most datasets. This shows the scalability and generalizability of the proposed approach, which is not influenced from the increase on the decaying dataset size. It is also important to note that the variations on the *NRMSE* obtained by *TBD-DP* between the datasets is mainly due to the different characteristics of each dataset.

Figure 12 examines the performance of the *TBD-DP* operator in terms of S and *NRMSE* when combined with three different ML models, namely, the traditional *Recurrent Neural Network (RNN)*, the *Gated Recurrent Unit (GRU)* [17] and the *Long Short Term Memory (LSTM)* that is finally adopted by the proposed approach. The results show that *TBD-DP* maintains a similar storage capacity for different learning models, with a slight increase (about 1%) when the LSTM model is used. In terms of *NRMSE*, however, the *TBD-DP*+LSTM combination clearly outperforms the other two combinations providing around 75% less error, on average.

Finally, Figure 13 examines how the number of neurons of the LSTM model influences the *TBD-DP*'s performance. The results support our previous observations on the scalability and generalizability of the proposed *TBD-DP* approach. The increase on the number of neurons slightly influences the *TBD-DP* in terms of storage capacity, since the required space slightly increases. This is reasonable since the increase on the number of neurons results in "bigger" models that require more disk space to be stored. The additional required space, however, is almost negligible compared to the disk space needed to store the actual data before decaying. In terms of *NRMSE*, the increase on the number of neurons does not influence the performance of the *TBD-DP* operator, since *NRMSE* remains almost the same while varying this control parameter in almost all datasets.

7.4 Experiment 3: Decaying Focus Experiments

In this experiment, the three decay focus methods are compared. Here it is important to revisit that: i) **FIFO-amnesia** decays f data based on the timestamp of the ingested tuples; ii) **UNIFORM-amnesia** decays f data based on a uniform random distribution. During the decaying procedure each tuple has the same probability to be decayed; and iii) **SPATIAL-amnesia** decays f data based on the spatial attribute of each record (e.g., cell id).

Figure 14 shows that decay focus *df* methods can improve the accuracy affecting the NRMSE. The SPATIAL-amnesia outperforms the FIFO-amnesia and UNIFORM-amnesia significantly. The results show that SPATIAL-amnesia has four times better accuracy than the other methods due to the fact that all the measurements were taken through the same telecommunication network and had similar characteristics. This confirm our initial hypothesis that we can improve the

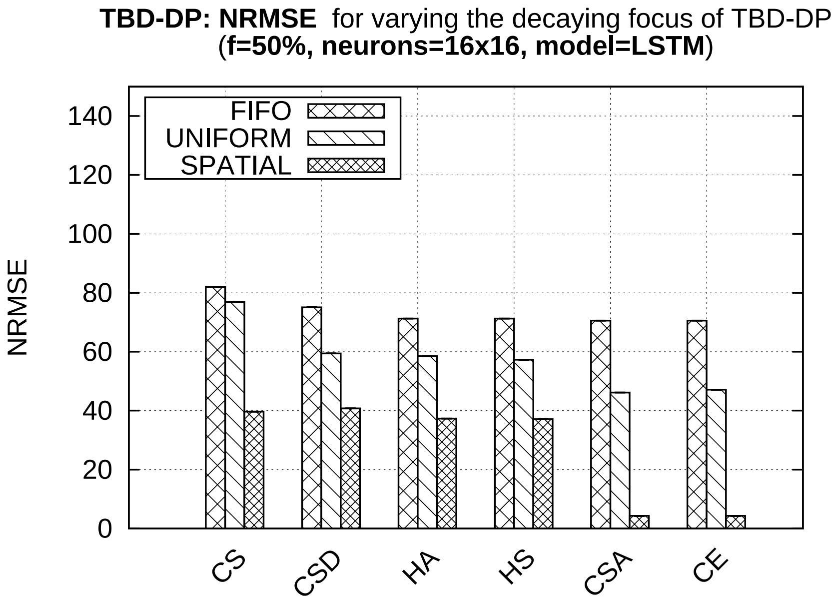

Fig. 14 Performance Evaluation: *TBD-DP* evaluation in terms of *NRMSE* varying the decaying focus for the decayed set of data in all datasets.

accuracy of our proposed *TBD-DP* operator using various decaying focus methods for domain-specific applications.

7.5 Experiment 4: *CTBD-DP* Experiments

We have divided the datasets into three consecutive batches (b_1, b_2, b_3) to evaluate the performance of our proposed *CTBD-DP* operator in a data streaming scenario where data arrive in batches. We chose to keep the FIFO-amnesia decaying focus method, as well as the same decay factor, number of neurons and model as in Experiment 1.

Figure 15 shows that the continuous learning can improve the accuracy by decreasing the *NRMSE*. The results show that *NRMSE* for b_3 is five time lower than b_1. As new batches are arriving in a streaming fashion, the retrieved model is re-trained allowing the accuracy to be improved. This is reasonable since the distribution of TBD has a repetitive pattern, as illustrated in Figure 8. It is also important to note that the variations on the final *NRMSE* obtained by *CTBD-DP* between the datasets is mainly due to the different characteristics of each dataset. Specifically, CE and CSA have a significant variation on the *NRMSE* obtain on b_1 with respect to b_3.

8 Conclusions

In this paper, we present two novel decaying operators for Telco Big Data (TBD), coined *TBD-DP* and *CTBD-DP*. *TBD-DP* relies on existing ML algorithms to ab-

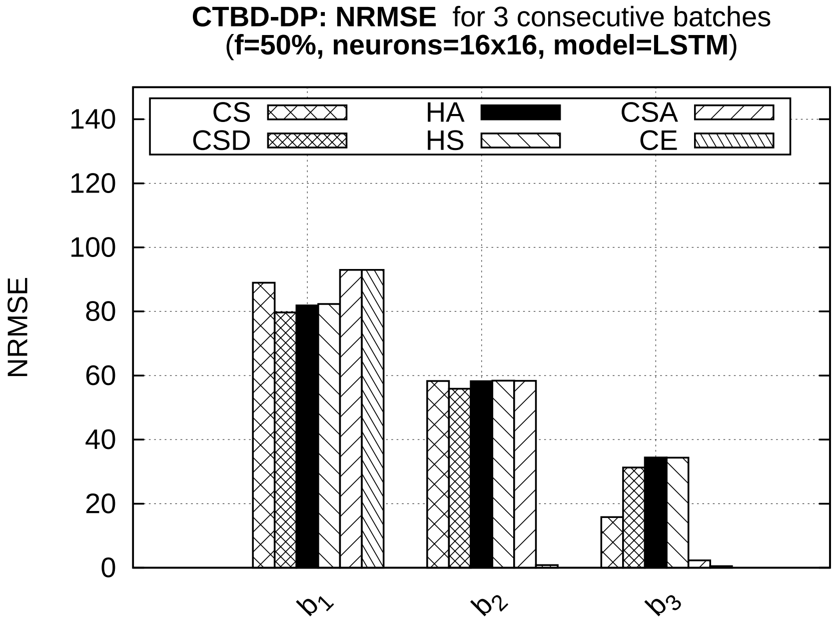

Fig. 15 Performance Evaluation: *CTBD-DP* evaluation in terms of *NRMSE* for three batches (b_1, b_2, b_3) for all datasets.

stract TBD into compact models that can be stored and queried when necessary. Our proposed *TBD-DP* operator has the following two conceptual phases: (i) in an offline phase, it utilizes a LSTM-based hierarchical ML algorithm to learn a tree of models (coined *TBD-DP* tree) over time and space; (ii) in an online phase, it uses the *TBD-DP* tree to recover data within a certain accuracy. *CTBD-DP* copes with TBD streams allowing the continuous decaying by utilizing the ability to restore the store models and continue the learning procedure. In our experimental setup, we measure the efficiency of the proposed operator using a ∼10GB anonymized real telco network trace and our experimental results in Tensorflow over HDFS are extremely encouraging as they show that *TBD-DP* saves an order of magnitude storage space while maintaining a high accuracy on the recovered data. Additionally, *CTBD-DP* is improving the accuracy as new batches are progressed keeping the storage space constant.

In the future, we aim to generalize data decaying operators beyond TBD into new domains (e.g., signals from other type of IoT). This task might give space to new ML algorithms. Additionally, we aim to theoretically derive the accuracy/efficiency bounds of our data postdiction framework. Finally, we plan to carry out an extensive experimental study that will focus solely on decaying of big data.

References

1. Abbasoğlu MA, Gedik B, Ferhatosmanoğlu H (2013) Aggregate profile clustering for telco analytics. Proc VLDB Endow 6(12):1234–1237, DOI

10.14778/2536274.2536284
2. Agarwal PK, Cormode G, Huang Z, Phillips J, Wei Z, Yi K (2012) Mergeable summaries. In: Proceedings of the 31st ACM SIGMOD-SIGACT-SIGAI Symposium on Principles of Database Systems, ACM, New York, NY, USA, PODS '12, pp 23–34, DOI 10.1145/2213556.2213562, URL http://doi.acm.org/10.1145/2213556.2213562
3. Agarwal S, Mozafari B, Panda A, Milner H, Madden S, Stoica I (2013) Blinkdb: Queries with bounded errors and bounded response times on very large data. In: Proceedings of the 8th ACM European Conference on Computer Systems, ACM, New York, NY, USA, EuroSys '13, pp 29–42, DOI 10.1145/2465351.2465355, URL http://doi.acm.org/10.1145/2465351.2465355
4. Barbará D, DuMouchel W, Faloutsos C, Haas PJ, Hellerstein JM, Ioannidis YE, Jagadish HV, Johnson T, Ng RT, Poosala V, Ross KA, Sevcik KC (1997) The new jersey data reduction report. IEEE Data Eng Bull 20(4):3–45, URL http://sites.computer.org/debull/97DEC-CD.pdf
5. Bhattacherjee S, Deshpande A, Sussman A (2014) Pstore: An efficient storage framework for managing scientific data. In: Proceedings of the 26th International Conference on Scientific and Statistical Database Management, ACM, New York, NY, USA, SSDBM '14, pp 25:1–25:12, DOI 10.1145/2618243.2618268, URL http://doi.acm.org/10.1145/2618243.2618268
6. Bhattacherjee S, Chavan A, Huang S, Deshpande A, Parameswaran A (2015) Principles of dataset versioning: Exploring the recreation/storage tradeoff. Proceedings of the VLDB Endowment 8(12):1346–1357
7. Bicer T, Yin J, Chiu D, Agrawal G, Schuchardt K (2013) Integrating online compression to accelerate large-scale data analytics applications. In: Parallel & Distributed Processing (IPDPS), 2013 IEEE 27th International Symposium on, IEEE, pp 1205–1216
8. Bouillet E, Kothari R, Kumar V, Mignet L, Nathan S, Ranganathan A, Turaga DS, Udrea O, Verscheure O (2012) Processing 6 billion cdrs/day: From research to production (experience report). In: Proceedings of the 6th ACM International Conference on Distributed Event-Based Systems, ACM, New York, NY, USA, DEBS '12, pp 264–267, DOI 10.1145/2335484.2335513
9. Braun L, Etter T, Gasparis G, Kaufmann M, Kossmann D, Widmer D, Avitzur A, Iliopoulos A, Levy E, Liang N (2015) Analytics in motion: High performance event-processing and real-time analytics in the same database. In: Proceedings of the 2015 ACM SIGMOD International Conference on Management of Data, ACM, New York, NY, USA, SIGMOD '15, pp 251–264, DOI 10.1145/2723372.2742783
10. Burtscher M, Ratanaworabhan P (2009) Fpc: A high-speed compressor for double-precision floating-point data. IEEE Transactions on Computers 58(1):18–31
11. Chaudhuri S, Das G, Narasayya V (2007) Optimized stratified sampling for approximate query processing. ACM Trans Database Syst 32(2), DOI 10.1145/1242524.1242526, URL http://doi.acm.org/10.1145/1242524.1242526
12. Cormode G, Garofalakis M, Haas PJ, Jermaine C (2012) Synopses for massive data: Samples, histograms, wavelets, sketches. Found Trends databases 4(1–3):1–294, DOI 10.1561/1900000004, URL http://dx.doi.org/10.1561/1900000004

13. Costa C, Zeinalipour-Yazti D (2018) Telco big data: Current state and future directions. In: Proceedings of the 19th IEEE International Conference on Mobile Data Management, IEEE Computer Society, ISBN: 978-1-5386-4133-0, June 27, 2018, Aalborg, Denmark, MDM'18, pp 11–12, DOI 10.1109/MDM.2018.00016
14. Costa C, Chatzimilioudis G, Zeinalipour-Yazti D, Mokbel MF (2017) Efficient exploration of telco big data with compression and decaying. In: 2017 IEEE 33rd International Conference on Data Engineering (ICDE), pp 1332–1343, DOI 10.1109/ICDE.2017.175
15. Costa C, Chatzimilioudis G, Zeinalipour-Yazti D, Mokbel MF (2017) Towards real-time road traffic analytics using telco big data. In: Proceedings of the International Workshop on Real-Time Business Intelligence and Analytics, BIRTE, Munich, Germany, August 28, 2017, pp 5:1–5:5, DOI 10.1145/3129292.3129296, URL http://doi.acm.org/10.1145/3129292.3129296
16. Costa C, Charalampous A, Konstantinidis A, Zeinalipour-Yazti D, Mokbel MF (2018) Decaying telco big data with data postdiction. In: 2018 19th IEEE International Conference on Mobile Data Management (MDM), pp 106–115, DOI 10.1109/MDM.2018.00027
17. Dey R, Salemt FM (2017) Gate-variants of gated recurrent unit (gru) neural networks. In: 2017 IEEE 60th International Midwest Symposium on Circuits and Systems (MWSCAS), pp 1597–1600, DOI 10.1109/MWSCAS.2017.8053243
18. Douglis F, Iyengar A (2003) Application-specific delta-encoding via resemblance detection. In: USENIX Annual Technical Conference, General Track, pp 113–126
19. Hochreiter S, Schmidhuber J (1997) Long short-term memory. Neural Comput 9(8):1735–1780, DOI 10.1162/neco.1997.9.8.1735, URL http://dx.doi.org/10.1162/neco.1997.9.8.1735
20. Hu X, Yuan M, Yao J, Deng Y, Chen L, Yang Q, Guan H, Zeng J (2015) Differential privacy in telco big data platform. Proc VLDB Endow 8(12):1692–1703, DOI 10.14778/2824032.2824067
21. Huang Y, Zhu F, Yuan M, Deng K, Li Y, Ni B, Dai W, Yang Q, Zeng J (2015) Telco churn prediction with big data. In: Proceedings of the 2015 ACM SIGMOD International Conference on Management of Data, ACM, New York, NY, USA, SIGMOD, pp 607–618, DOI 10.1145/2723372.2742794
22. Iyer AP, Li LE, Stoica I (2015) Celliq: Real-time cellular network analytics at scale. In: Proceedings of the 12th USENIX Conference on Networked Systems Design and Implementation, USENIX Association, Berkeley, CA, USA, NSDI'15, pp 309–322
23. Kersten ML (2015) Big data space fungus. In: CIDR 2015, Seventh Biennial Conference on Innovative Data Systems Research, Asilomar, CA, USA, January 4-7, 2015, Online Proceedings
24. Kersten ML, Sidirourgos L (2017) A database system with amnesia. In: CIDR
25. Krishna K, Jain D, Mehta SV, Choudhary S (2018) An lstm based system for prediction of human activities with durations. Proc ACM Interact Mob Wearable Ubiquitous Technol 1(4):147:1–147:31, DOI 10.1145/3161201, URL http://doi.acm.org/10.1145/3161201
26. LaChapelle C (2016) The cost of data storage and management: where is the it headed in 2016? URL http://www.datacenterjournal.com/cost-data-storage-

management-headed-2016/
27. Laiho J, Wacker A, Novosad T (2006) Radio Network Planning and Optimisation for UMTS. John Wiley & Sons
28. Lakshminarasimhan S, Shah N, Ethier S, Klasky S, Latham R, Ross R, Samatova NF (2011) Compressing the incompressible with isabela: In-situ reduction of spatio-temporal data. In: European Conference on Parallel Processing, Springer, pp 366–379
29. Luo C, Zeng J, Yuan M, Dai W, Yang Q (2016) Telco user activity level prediction with massive mobile broadband data. ACM Trans Intell Syst Technol 7(4):63:1–63:30, DOI 10.1145/2856057
30. Savitz E (2012) Forbes magazine. URL https://goo.gl/eM1uwV, [Online; April 16, 2012]
31. Schendel ER, Jin Y, Shah N, Chen J, Chang CS, Ku SH, Ethier S, Klasky S, Latham R, Ross R, et al. (2012) Isobar preconditioner for effective and high-throughput lossless data compression. In: 2012 IEEE 28th International Conference on Data Engineering, IEEE, pp 138–149
32. Sidirourgos L, Martin, Boncz P (2011) Sciborq: Scientific data management with bounds on runtime and quality. In: In Proc. of the Intl Conf. on Innovative Data Systems Research (CIDR, pp 296–301
33. Soroush E, Balazinska M (2013) Time travel in a scientific array database. In: Data Engineering (ICDE), 2013 IEEE 29th International Conference on, IEEE, pp 98–109
34. Wei Z, Luo G, Yi K, Du X, Wen JR (2015) Persistent data sketching. In: Proceedings of the 2015 ACM SIGMOD International Conference on Management of Data, ACM, New York, NY, USA, SIGMOD '15, pp 795–810, DOI 10.1145/2723372.2749443, URL http://doi.acm.org/10.1145/2723372.2749443
35. Yan H, Ding S, Suel T (2009) Inverted index compression and query processing with optimized document ordering. In: Proceedings of the 18th international conference on World wide web, ACM, pp 401–410
36. You LL, Pollack KT, Long DD, Gopinath K (2011) Presidio: a framework for efficient archival data storage. ACM Transactions on Storage (TOS) 7(2):6
37. Yuan M, Deng K, Zeng J, Li Y, Ni B, He X, Wang F, Dai W, Yang Q (2014) Oceanst: A distributed analytic system for large-scale spatiotemporal mobile broadband data. Proc VLDB Endow 7(13):1561–1564, DOI 10.14778/2733004.2733030
38. Zeng K, Agarwal S, Dave A, Armbrust M, Stoica I (2015) G-ola: Generalized on-line aggregation for interactive analysis on big data. In: Proceedings of the 2015 ACM SIGMOD International Conference on Management of Data, ACM, New York, NY, USA, SIGMOD '15, pp 913–918, DOI 10.1145/2723372.2735381, URL http://doi.acm.org/10.1145/2723372.2735381
39. Zhang S, Yang Y, Fan W, Lan L, Yuan M (2014) Oceanrt: Real-time analytics over large temporal data. In: Proceedings of the 2014 ACM SIGMOD International Conference on Management of Data, ACM, New York, NY, USA, SIGMOD '14, pp 1099–1102, DOI 10.1145/2588555.2594513
40. Zhu F, Luo C, Yuan M, Zhu Y, Zhang Z, Gu T, Deng K, Rao W, Zeng J (2016) City-scale localization with telco big data. In: Proceedings of the 25th ACM International on Conference on Information and Knowledge Management, ACM, New York, NY, USA, CIKM, pp 439–448, DOI 10.1145/2983323.2983345