Pi-CEP: Predictive Complex Event Processing using Range Queries over Historical Pattern Space

Syed Gillani, Abderrahmen Kammoun, Kamal Singh, Julien Subercaze, Christophe Gravier, Jacques Fayolle, Frederique Laforest

Pi-CEP: Predictive Complex Event Processing using Range Queries over Historical Pattern Space

Syed Gillani, Abderrahmen Kammoun, Kamal Singh, Julien Subercaze,
Christophe Gravier, Jacques Fayolle and Frédérique Laforest
Univ Lyon, UJM-Saint-Etienne, CNRS, Laboratoire Hubert Curien UMR 5516, F-42023
Saint Étienne, France
firstname.lastname@univ-st-etienne.fr

***Abstract*—*Predictive* Complex Event Processing (CEP) constitutes the next phase of CEP evolution and provides future predictive states of the partially matched complex sequences. In this paper, we demonstrate our novel predictive CEP system and show that this problem can be solved while leveraging existing data modelling, query execution and optimisation frameworks. We model the predictive detection of events over an N-dimensional historical matched sequence space. Hence, a predictive set of events can be determined by answering the range queries over the historical sequence space. In order to take advantage of range search over 1-dimensional data structures, we transform the N-dimensional space into 1-dimension using *space filling z-order curve*. We propose a compressed index structure to store 1-dimensional data and execute customised range query techniques. Furthermore, we propose an approximate summarisation technique, over the historical space of top-k most infrequent range queries, to cater catastrophic forgetting of older matches. Two real-world datasets are used to demonstrate the feasibility of our proposed techniques. We demonstrate that our system can efficiently predict complex events and it equips a user-friendly interface to fulfil the requirements of user-computer interaction in a real-time.**

***Index Terms*—Complex Event Processing, Prediction, Z-curve, Range Queries**

I. Introduction

With the proliferation of applications producing data in a streaming manner, Complex Event Processing (CEP) has become a de facto standard to support use cases ranging from fraud detection, stock market analysis to health care analytics [17]. Given a pattern query Q, a stream $S = \{e_1, e_2, \dots\}$ – where each event forms a tuple $e = (A, \tau)$ of attribute values $A = \{a_1, a_2, \dots, a_l\}$ associated with a timestamp τ – the objective of the CEP system is to detect chronologically ordered sequences of events, each of the form $\overrightarrow{e}_f = \langle e_1, \dots, e_m \rangle$ $m > 0$, that occur in the event stream and are correlated based on the values of the attributes and defined temporal operators in the pattern query Q. The pattern query Q over S is evaluated in a progressive way. That is, partial matches $\overrightarrow{e}_p = \langle e_1, \dots, e_i \rangle$, where $i < m$, are formed before a full match is detected.

Today, the analytics are moving towards a model of proactive computing [7] and the CEP also requires a paradigm shift towards proactive and predictive computations. That is, given a partial match $\overrightarrow{e}_p$, a *predictive CEP* provides the possible future events of the partially matched sequences which can turn it into a full match $\overrightarrow{e}_f$. This would enable the users to mitigate or eliminate undesired future events or states and identify future opportunities.

The problem of predictive CEP shows remarkable similarities with the *sequence pattern mining and prediction*. In this context, a large body of sequence prediction models have been proposed including: Prediction By Partial Matching (PPM) [5], All-K-Order-Markov [13] and Probabilistic Suffix Tree [3]. These models are based on the Markov property and suffer from the catastrophic forgetting of older sequences, where only the k recent items of training sequences are used to perform prediction: increasing k often induces a very high state complexity and consequently such techniques become impractical for many real-life applications [9].

This paper studies the problem of sequence prediction in the context of CEP systems. However, our techniques are also relevant in the context of general multidimensional sequence prediction. We propose a new view of this problem with the main aim to push the real-time predictive CEP capabilities, to the database layer, so as to take advantage of and extend existing data structures, query execution and optimisation techniques.

Our Contributions: Our approach leverages two key observations: (a) the historical matches can give an "expert" view of the future matches; (b) summarising the older matches according to their observed importance can avoid the catastrophic forgetting, while operating in the main memory for a real-time response.

To administer the first observation, we propose a novel N dimensional (N-D) historically matched sequence space $\mathcal{H}_s$. Based on this, we design an index structure that leverages the embedding of *fractal-based space-filling curve*, the *Z-order curve* [12], to map the coordinates of the N-D space into a 1-D space, while preserving the locality of points. In order to query 1-D points, we propose a novel range query algorithm that caters the unbalanced nature of the partial matches, and locates the nearby points for predictive analysis.

To administer the second observation, since the index size expands proportionally to the number of matched sequences, we need to compress the older matches while preserving their importance. To summarise older points in $\mathcal{H}_s$, we first gather the points covered by *top-k* most infrequent range queries in a streaming manner, second, we use the *weighted average mean*

to summarise the points that are closer to each other. The weights of the points are determined by their frequency in $\mathcal{H}_s$. This not only provides an efficient way for summarising older data but also offers an efficient way of keeping the older matched sequences according to their importance.

Our experimental evaluation over two real-world datasets shows the significance of our indexing, querying and summarising techniques. Our system outperforms the competitor by a considerable margin in terms of accuracy and performance.

Integrating the contributions specified above, we demonstrate a system for predictive CEP, called Pi-CEP (Predictive Complex Event Processing). It provides the aforementioned complementary functionalities and can be integrated with existing general purpose CEP systems. Additionally, we provide a user-friendly interface for users to interact and visualise the results. Moreover, we provide customised search components to meet the demands of advanced users.

This paper is organised as follows. Section II presents the related work, then our approach is presented in Section III followed by section IV which shows some results. Finally, Section V describes our demonstration and Section VI concludes the paper.

II. Related Works

Sequence Pattern Mining and Prediction: A large number of sequence prediction methods [3], [5], [9], [13] come from the field of temporal/time-series pattern mining, where patterns are defined using association rules or as frequent episodes. These methods employ variants of decision trees and probabilistic data structures. Most of them are based on the Markov property and suffer from the problem of forgetting older models and high computation costs. Recently, two methods CPT and CPT+ [9] are proposed that keep all the data from the history in a compressed format and offer increased accuracy. It is based on the prefix tree (aka trie) data structure and only support one dimensional sequences. Furthermore, these models are not optimised for streaming applications, where the training dataset is unknown; a large number of events arrive with a high rate; events occur at random arrival times rather than at the regular tick-tock intervals of traditional time series and a real-time response is required.

Machine Learning-based Predictions: Further directions for tackling this problem are incremental (online) machine learning algorithms that learn incrementally over event streams, such as Support Vector Machines, recurrent Neural Networks, Bayesian Networks [1], [8], [11]. For these algorithms, the classifiers are updated each time a new training instance is found to provide a predictive response. The main disadvantages of these algorithms in the context of CEP are as follows: (i) they are use case specific and require considerable efforts (and training datasets) to model each dataset, e.g. a model can either be based on the *recent* history or is *updatable* based on the older values; (ii) they do not provide any performance guarantees in terms of error bounds, which brings additional difficulties on actions such as performance tracking and regular maintenance: since the learned parameters keep on changing dynamically [11]. Our approach goes beyond the simple sequence prediction, and prediction is made on the multidimensional and variable length sequences.

III. Our Approach

We are given a collection $\{\overrightarrow{e}_{f_1}, \overrightarrow{e}_{f_2}, \ldots, \overrightarrow{e}_{f_s}\}$ of fully matched sequences arrived in a chronological manner. Then for a partially matched sequence $\overrightarrow{e}_p$, our task is to predict future events that can turn $\overrightarrow{e}_p$ into $\overrightarrow{e}_f$ using the universe of fully matched sequences. That is, if m is the length of the fully matched sequence then for a partially matched sequence $\overrightarrow{e}_p = \langle e_1, \ldots, e_i \rangle$ $(i < m)$, we would like to predict the future events from e_i to e_m. To attain this, we model our solution based on an N-D historical matched event database called *historical space* $\mathcal{H}_s$. Hence, using $\mathcal{H}_s$, we can employ range queries for the partially matched sequences to determine the predictive events.

Let $\mathcal{H}_s = \{A_1 \times A_2 \times, \ldots, A_n\}$ be an N-D *lattice* for the universe of fully matched sequences, where an n-tuple $X = (x_1, x_2, \ldots, x_n)$ defines a point in $\mathcal{H}_s$ and $x_i \in A_i \; \forall i \in n$. Then with the arrival of a partially matched sequence $\overrightarrow{e}_p$, we answer *N-D range queries* on $\mathcal{H}_s$, which in turn is detailed later in this section. The points lying within the range queries or its neighbours form the predictive events for a partially matched sequence $\overrightarrow{e}_p$.

A large number of works (such as Quad Trees, KD-Trees, etc.) [10], [12], [16] have been proposed for encoding N-D space. Considering the large number of dimensions in our context, and the effectiveness of space-filling curves in spatial domains [16], we use the Z-order curve [12] for N-D space encoding. The Z-order curve preserves the proximity properties among the points, while leveraging the effectiveness of linear data structures (such as B+tree) for range queries. The construction of the Z-order curve, i.e. Z-values generations, is accomplished by the simple process of *bit-shuffling* and this is one of its main advantages over other space-filling curves. An example of a 2D Z-curve and bit shuffling process is shown in Figure 1.

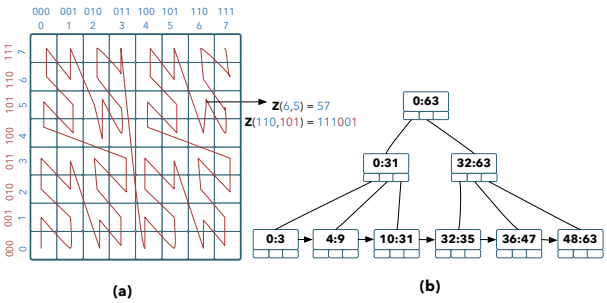

Fig. 1: (a) Z-order curve for a 2-D space, (b) Z-value encoded B+tree

We leverage the Z-order curve with B+tree for our indexing scheme [14]. The nodes of the B+tree are sequentialised using Z-order curve to retrieve a given region efficiently (see Figure 1). In order to provide the compact representation

of encoded dimensions, we employ compressed bitmaps [4] to store large Z-values and to compress the sequence of homogeneous 1's and 0's within the Z-values. This not only provides a good compression ratio but also simplifies the comparison of Z-values by using fast bitwise operations over the bitmaps which are supported by the hardware. The Z-value, the *timeline* of the Z-value and the number of time it appears in the history form the basic of our indexing scheme. We call it *temporal Z-value*.

Definition 1. *(Temporal Z-value). A temporal Z-value $\mathcal{Z}_\tau$ = ($\mathbf{Z}, T, f$) contains a Z-value $\mathbf{Z}$, a set of timestamps T to denote its timeline, and a frequency f times $\mathbf{Z}$ appears in the historical space $\mathcal{H}_s$.*

The essence of the temporal Z-value is that, due to the iterative nature of the underlying fractal, it imposes the required sequential order of matched sequences on $\mathcal{H}_s$. One other attractive feature of our indexing technique is that $\mathcal{H}_s$ can be efficiently stored and retrieved from the main-memory due to the linear properties of B+trees.

A. Querying Historical Space for Prediction

Pre-processing Stage: We first consider the case when all the matched sequences in $\mathcal{H}_s$ are of equivalent length (i.e. of same dimensions). In the pre-processing stage, we first construct the range queries and their corresponding Z-values. For this task, we maintain an inverted index to record *minimum* and *maximum* values of all dimensions in $\mathcal{H}_s$. That is, if there are d_p known dimensions in $\overrightarrow{e}_p$ and d_m is the expected total dimensions of the $\overrightarrow{e}_f$, the minimum and maximum values of the range query are defined as follows:

$$X_{min} = \left(x_1, \ldots, x_{d_p}, min(x_{d_{p+1}}), \ldots, min(x_{d_m})\right)$$
$$X_{max} = \left(x_1, \ldots, x_{d_p}, max(x_{d_{p+1}}), \ldots, max(x_{d_m})\right),$$

where $\forall x_i \in A_i, min(x_i) \leq x_i \wedge max(x_i) \geq x_i$ and note that $x_1, \ldots, x_{d_p}$ are already known as they correspond to the already known points in the partially matched sequence. The range query points are then mapped onto the Z-values, i.e. $\mathbf{Z}_{min} = Zval(X_{min})$ and $\mathbf{Z}_{max} = Zval(X_{max})$. These range query points are used to traverse in the B+tree. Furthermore, since both have the same values for known dimensions, any of them can be used to determine if a point in $\mathcal{H}_s$ can be enclosed by the range query points.

The case of determining range queries when $\mathcal{H}_s$ contains variable length (dimensions) matched sequences is different. For example, if $\overrightarrow{e}_{f_1} = \langle e_1, e_2, e_3 \rangle$ and $\overrightarrow{e}_{f_2} = \langle e_1, e_2, e_3, e_4 \rangle$ are two matched sequences in $\mathcal{H}_s$, a single range query will not cover both of them. One, and rather expensive, solution for this problem would be to create a set of range queries each covering the matched sequences of specific dimensions. We, on the other hand, employ an adaptive technique during the post-processing stage, while searching for the points covered by a range query. Given the known dimensions in the range query, we exclusively check if a point having the variable dimensions can be covered by the known dimensions or not.

Post-processing Stage: In this stage, we search over the nodes of the B+tree that are intersected by a range query, which is constructed in the pre-processing stage. This is done by checking the bits, of known dimensions, of tree node's Z-value $\mathbf{Z}$ and either the $\mathbf{Z}_{min}$ or $\mathbf{Z}_{max}$ value of the range query that we call $\mathbf{Z}_r$. Let g be a function mapping the Z-value to a bitstring (or binary number), we use $g_t^s(z)$ (small caps for Z-value's bitstring for brevity) to denote all the t most significant bits of $g(z)$ each skipped by s bits; e.g. $g_2^1(z)$ = 1010 for z = 101101. We say that for two bitstrings z_1 and z_2 , z_1 is a t_1-prefix-s_1 of z_2 , iff z_1 is identical to all the highest t_1 bits in z_2 each skipped by s_1 bits ; e.g. **1010** is a 2-prefix-1 of **10**1**10**1. Hence for a tree node's Z-value bitstring $g(z)$ and given query range bitstring $g(z_r)$, we compute $g_t^s(z)$, $g_t^s(z_r)$ and if z is a t-prefix-s of z_r, it shows that z is enclosed by the given range query. The values of t and s are the number of known and unknown dimensions in the given $\overrightarrow{e}_p$ respectively. Thus, using fast bitwise operations, we can easily find the contender set of points in $\mathcal{H}_s$ for prediction. Figure 2 shows the computation of $g_t^s(z)$ and the range query point $g_t^s(z_r)$.

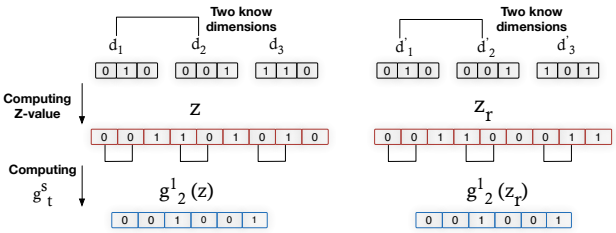

Fig. 2: Z-value encoding and calculation of g_t^s over a point z in $\mathcal{H}_s$ (left) and the range query point z_r (right) with two known dimensions.

The same g_t^s function is also used to match variable length sequence points in $\mathcal{H}_s$ and the range query point. However, in this case the values of t and s are adjusted both for the points z and z_r. For instance, if there are two unknown dimensions and one known dimension in z and there are three unknown dimensions and one known dimension in z_r, the functions are as follows: $g_1^2(z)$ and $g_1^3(z_r)$. The skipping factor caters the variability of length between points in $\mathcal{H}_s$ and the range query point.

Reaching out to the Neighbours: The comparison of range query and the points in $\mathcal{H}_s$ is initially strict for each prediction task. For a given partial match sequence $\overrightarrow{e}_p$, if the post-processing stage cannot find similar sequences to generate a prediction, we assume that the partial match contains some noise and the post-processing stage is dynamically relaxed. That is, first we get the neighbouring points of the range query Z_r that lies under the same B+tree node; second we check the similarity on sequence of most significant bits (MSB) of the neighbouring point's bitstrings $g_t^s(z)$ and $g_t^s(z_r)$, we call it MSB similarity. Let $h(g_t^s(z), g_t^s(z_r))$ be a function that determines the length of the similar MSB of two points; e.g. $h(g_t^s(z_1), g_t^s(z_2))$ = 4 for $g_t^s(z_1)$ =**0110**00 and $g_t^s(z_2)$ = **0110**11. Given two points z_i and z_j, if $h(g_t^s(z_i), g_t^s(z_r)) < h(g_t^s(z_j), g_t^s(z_r))$, it shows that z_j is closer to z_r compared

with z_i. Note that the above similarity measures are designed by considering the nature of Z-order curves and techniques such as *Hamming distance* would not be appropriate in our context. Furthermore, similarity measures are used to add the $\pm$ error margins. The set of neighbours with the highest MSB similarity according to the defined error margins are then chosen for the predictive response.

B. Summarisation of Historical Space Points

The process of summarisation is divided into the following two tasks: (1) continuously evaluate and update top-k most infrequent range queries generated by the system; (2) approximate summarisation of the older points in $\mathcal{H}_s$, covered by the top-k most infrequent range queries, that are closer to each other in $\mathcal{H}_s$. This procedure is incremental and applied continuously over the system's life-time. The first task is well-studied and we employ a sketch and a heap structure to provide approximate frequencies and ordering of range queries. Our sketch stems from a count-min sketch [6], but we modify it to incorporate range query information. Once a defined window expires for a pattern query, we extract the top-k most infrequent range queries in the sketch.

For the second task, recall from earlier, the Z-order curve preserves the proximity of points in $\mathcal{H}_s$. Hence, points under the same parent node in B+tree are inherently closer to each other. We use an approximate temporal Z-value $\hat{\mathcal{Z}}_\tau$ to summarise points under the same B+tree node which fall in the top-k most infrequent range queries. Our approximation technique is based on the *weighted linear combinations* (WLC) of Z-values: while other options exist, WLC has a strong history in successfully modelling the irregular and continuous characteristics of the data according to the observed weights. We compute $\hat{\mathcal{Z}}^n_\tau$ as a WLC of points closer to each other, i.e. under the same B+tree node:

$$\hat{\mathcal{Z}}^n_\tau = \sum_{i\,=\,0}^{n} w_i \cdot \mathcal{Z}^i_\tau, \;\; and \;\; w_i = \frac{f(\mathcal{Z}^i_\tau)}{\sum_{i=0}^{n} f(\mathcal{Z}^i_\tau)}$$

where $0 < w_i \leq 1$ denotes the weight given to a $\mathcal{Z}^i_\tau$. It is calculated from the set of points to be summarised together: we assign higher weights to the points having high frequency f in $\mathcal{H}_s$ and $f(\mathcal{Z}^i_\tau)$ is the frequency of the i^{th} point to be summarised.

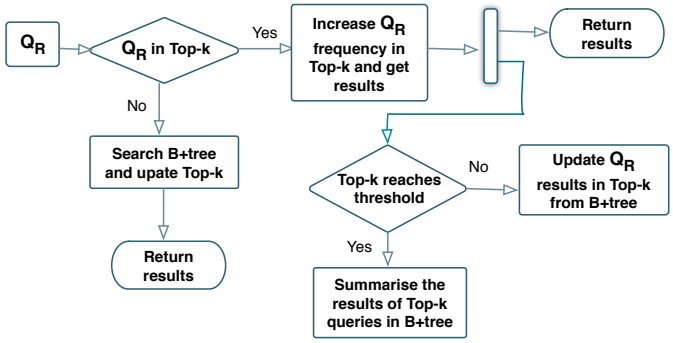

Fig. 3: Range Query Processing with Summarisation

The complete evaluation of range query with summarisation is described in Figure 3. Given an incoming range query Q_R, we first check whether Q_R exists in top-k or not. If not, we employ the usual range query search in B+tree and update the top-k for Q_R. Otherwise, we update the frequency of Q_R in top-k and return the result of Q_R. Furthermore, we check if the frequency of the top-k range queries has reached the defined threshold. If so we generate the summary of range queries points and update the B+tree index. Otherwise, we update the results of Q_R, if any, from the B+tree. Finally, the result generator (in Figure 6) processes the results from the range query processor and sends it to the user interface in an appropriate format.

IV. Experimental Evaluation

In this section, we evaluate the performance and precision of our proposed techniques.

Experiment Setup: Our algorithms are implemented using Java and evaluated on an Intel Xeon E3 1246v3 processor with 32GB of main memory and a 256Gb PCI Express SSD. It runs a 64-bit Linux 3.13.0 kernel with Oracle's JDK 8u112. For robustness, we performed 10 independent runs and we report the median values.

Datasets and CEP Queries: We use two real-world datasets for evaluation. All the datasets are first processed using SASE+ [17] CEP system to generate partially matched sequences (PM) and fully matched sequences (FM) streams. These streams are then chronologically fed to our system. The two real-world datasets include: *Activity Dataset* [15] (1 million FM, 500K PM) and *Credit Card Transactions Dataset* [2] (500K FM, 600K PM). The queries to generate these datasets are as follows: (i) *cardiac arrhythmia detection*: if the heart rate gradually increases until it doubles compared to the first measurement despite passive physical activity; (ii) *credit card fraud*: if a credit card is used to take large sums of money in a sequence at different locations within a time window.

Accuracy Metrics: We provide the accuracy metrics as follows: $\frac{\text{\# of correctly predicted PM}}{\text{\# of PM}}$. We keep the predicted PM until a FM arrives. Once a FM is detected, we compare if the predicted PM satisfies the FM. Furthermore, $k = 10$ is empirically chosen for all the experiments and an error of $\pm 5\%$ is considered for the correctly predicted PM.

Precision of Prediction with Summarisation: The first question we investigate is "*How useful is the summarisation process w.r.t deleting older fully matched sequences?*" This measures the effect of forgetting older matched sequences on prediction. Figure 4(a) and (b) shows the prediction accuracy of both datasets by varying the matched sequences. To showcase the effectiveness of summarisation, we forget the older matches after a window expires. From Figure 4, our summarisation technique results in better accuracy as the time passes, since older values aid in predicting the future matches. However, forgetting the history with the expiration of a window results in reduced precision and only recently matched sequences are used for prediction. Furthermore, since

similar matched sequences repeat for the activity dataset (Figure 4(b)), its evaluation provides better accuracy measures.

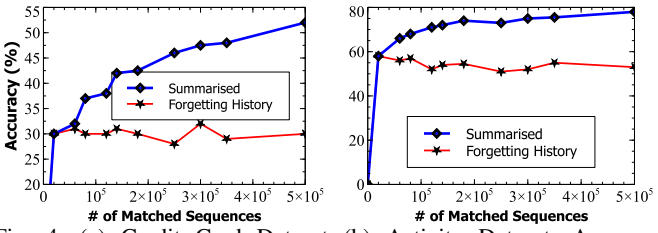

Fig. 4: (a) Credit Card Dataset (b) Activity Dataset: Accuracy comparison of prediction for the number of matched sequence;

Comparison with other Techniques: Next, we investigate "*How our techniques are compared to the existing sequence prediction techniques (i.e. CPT+ [9]) both in terms of performance and accuracy?*" For this task, we extend CPT+ to work on streams, i.e. FM sequences are inserted in a streaming fashion and PM sequences are used to predict the sequences in a streaming manner. Note that CPT+ only supports sequences with one-dimension, therefore, we use the Credit Card Transactions Dataset with 1-dimensional sequences. Figure 5(a) shows the accuracy comparison of both systems, while Figure 5(b) shows the performance of inserting matched sequences and querying predictive sequences, while varying the number of fully matched sequences. From Figure 5, our system outperforms CPT+ both in terms of accuracy and performance. The reasons are summarised as follows: (i) CPT+ identifies frequent sequences in training phase for efficient compression and prediction, this however is not efficient in streaming settings; (ii) due to the non-linear nature of CPT+ trie structure, it requires m comparisons for a sequence of length m and for n distinct sequences, the cost is $O(m \times n)$: B+tree requires $O(log_b\ n)$, with branching factor b, for the lookup and insertion using fast bitwise operations.

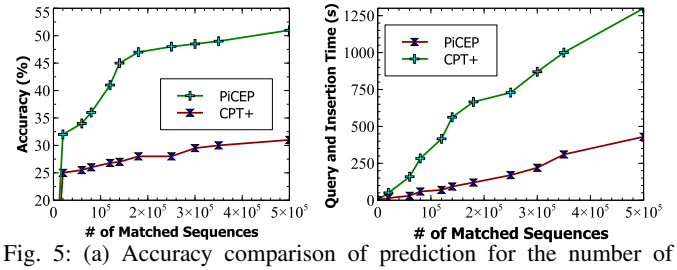

Fig. 5: (a) Accuracy comparison of prediction for the number of matched 1-dimensional sequences (Credit Card Dataset); (b) Execution time in seconds for the insertion and prediction using Credit Card Dataset

V. Demonstration

In this section, we would like to demonstrate how Pi-CEP provides predictive analysis using aforementioned techniques and customised parameters.

System Architecture for Demonstration: As mentioned already, the core functionality of Pi-CEP is implemented in Java, while the user interface is implemented in HTML and JavaScript. The high-level system architecture of the Pi-CEP is shown in Figure 6. The event streams and pattern queries are fed to the CEP engine, while customised parameters for the range queries, i.e. value of k are fed to the range query processor. For our current implementation, we are using the SASE+ CEP engine [17]. The CEP engine produces the fully and partial matches that are delegated to the Z-value generator and range query generator respectively. The Z-values of the fully matched sequences are stored in B+tree index, while generated range queries are delegated to the range query processor.

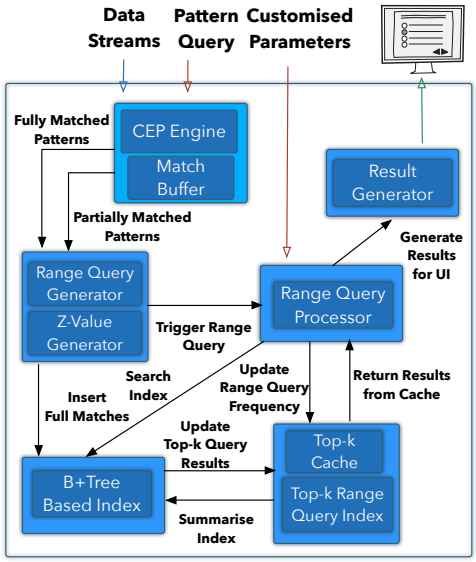

Fig. 6: System Architecture of Pi-CEP

User Interface: The user interface is shown in Figure 7. It is used to provide the interactive results of fully matched patterns and predictive events for the partially matched patterns. It consists of three main parts: the control panel (left); the graph display panel (bottom right); and a fully/predictive matched pattern panel (top right). The control panel is used to let users specify the input pattern queries, select datasets and customised parameters for the range queries, such as k value. The graph display panel shows two real-time graphs: the fully matched and predictive patterns; and the change in the size of the underlying B+tree index with the arrival of events and after the summarisation process. The matched pattern panel shows the real-time matched set of events for a defined pattern and the predictive events provided by our system. We use different coloured schemes for both of them to visualise it in an aesthetic fashion. Moreover, the system can be paused and resumed to compare each produced result.

We plan to demonstrate the three main features of Pi-CEP. (1) Our system can provide predictive events. (2) Pi-CEP can efficiently evaluate and update top-k most infrequent range queries generated by the system and then summarise the older points in $\mathcal{H}_s$. The summarisation measures can be customised using different values of k. (3) Pi-CEP equips

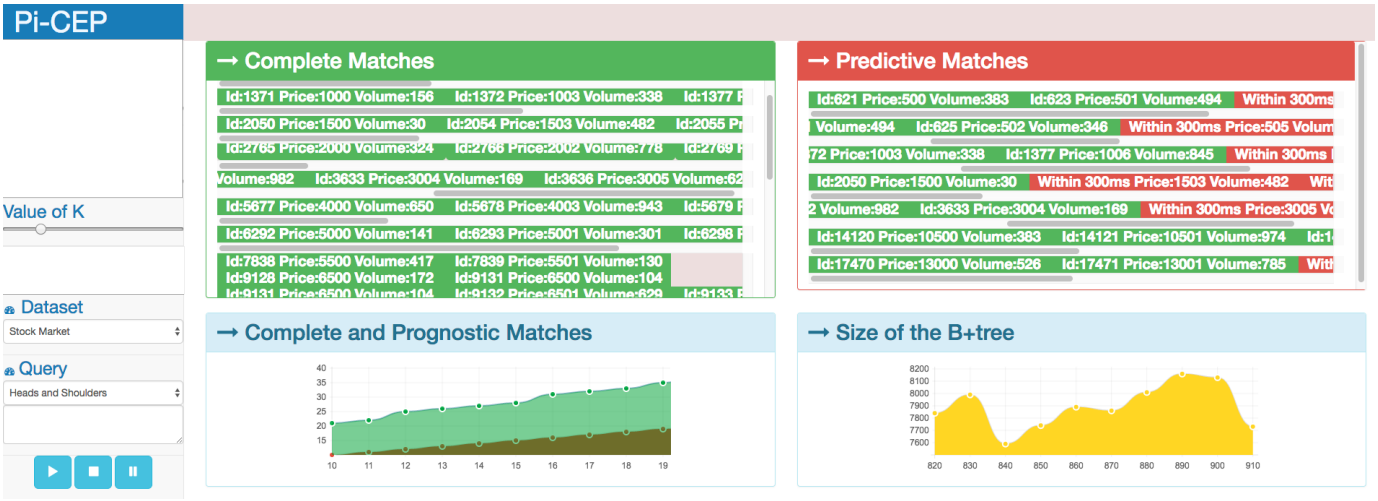

Fig. 7: Interface of Pi-CEP

a user-friendly interface to fulfil user-computer interaction requirements in a real-time. During the demonstration, we will explain the details and nuances of our system, discuss the presented statistics and the decision taken by the system that yielded these outcomes. In particular, we will discuss why certain behaviours are shown by our system for different pattern queries and explain the effects of various customised parameters.

VI. Conclusion

Pi-CEP facilitates the predictive analysis over partially matched sequences. It employs efficient indexing techniques to store multi-dimensional fully matched sequences and extract the predictive matches using customised range queries. Furthermore, it provides efficient techniques to summarise the older fully matched sequences based on their importance. This enables us to keep track of both recent and older history for predictive analysis. We demonstrate these characteristics of Pi-CEP using an interactive interface.

Acknowledgements. This work is partially supported by ITEA 3 project Water-M with the funding from DGE France.

References

[1] M. Akdere, U. Çetintemel, and E. Upfal. Database-support for continuous prediction queries over streaming data. *VLDB*, 2010.

[2] A. Artikis, N. Katzouris, I. Correia, C. Baber, N. Morar, I. Skarbovsky, F. Fournier, and G. Paliouras. A prototype for credit card fraud management: Industry paper. In *DEBS*, pages 249–260, 2017.

[3] R. Begleiter, R. El-Yaniv, and G. Yona. On prediction using variable order markov models. *J. Artif. Int. Res.*, 22(1):385–421, Dec. 2004.

[4] S. Chambi, D. Lemire, O. Kaser, and R. Godin. Better bitmap performance with roaring bitmaps. *CoRR*, 2014.

[5] J. G. Cleary and I. H. Witten. Data compression using adaptive coding and partial string matching. *IEEE Transactions on Communications*, 32(4):396–402, 1984.

[6] G. Cormode. Sketch techniques for approximate query processing. In *Foundations and Trends in DB*, 2011.

[7] Y. Engel and O. Etzion. Towards proactive event-driven computing. In *DEBS*, pages 125–136, 2011.

[8] L. J. Fülöp and Beszédes. Predictive complex event processing: A conceptual framework for combining complex event processing and predictive analytics. In *BCI*, 2012.

[9] T. Gueniche, P. Fournier-Viger, R. Raman, and V. S. Tseng. CPT+: Decreasing the time/space complexity of the compact prediction tree. In *PAKDD*, pages 625–636, 2015.

[10] D. Hilbert. *Ueber stetige abbildung einer linie auf ein flachenstuck. Mathematische Annalen*. 1891.

[11] M. Mohri, A. Rostamizadeh, and A. Talwalkar. *Foundations of Machine Learning*. MIT Press, 2012.

[12] G. M. Morton. A computer oriented geodetic data base; and a new technique in file sequencing. In *IBM*, 1966.

[13] J. Pitkow and P. Pirolli. Mining longest repeating subsequences to predict world wide web surfing. In *USENIX Symposium on Internet Technologies and Systems*, pages 13–13, 1999.

[14] F. Ramsak, V. Markl, R. Fenk, M. Zirkel, K. Elhardt, and R. Bayer. Integrating the ub-tree into a database system kernel. In *VLDB*, 2000.

[15] A. Reiss and D. Stricker. Creating and benchmarking a new dataset for physical activity monitoring. In *PETRA*, pages 40:1–40:8, 2012.

[16] H. Samet. *Foundations of Multidimensional and Metric Data Structures*. Morgan Kaufmann Publishers Inc., 2005.

[17] E. Wu, Y. Diao, and S. Rizvi. High-performance complex event processing over streams. In *SIGMOD*, 2006.