DLACEP: A Deep-Learning Based Framework for Approximate Complex Event Processing

Adar Amir
Technion, Israel Institute of Technology
Haifa, Israel
adaramir@cs.technion.ac.il

Ilya Kolchinsky
Technion, Israel Institute of Technology
Haifa, Israel
ikolchin@cs.technion.ac.il

Assaf Schuster
Technion, Israel Institute of Technology
Haifa, Israel
assaf@cs.technion.ac.il

ABSTRACT

Complex event processing (CEP) is employed to detect user-specified patterns of events in data streams. CEP mechanisms operate by maintaining all sets of events that can potentially be composed into a pattern match. This approach can be wasteful when many of the sets do not participate in an actual match and are therefore discarded.

We present DLACEP, a novel framework that fuses deep learning with CEP to efficiently extract complex pattern matches from streams. To the best of our knowledge, this is the first time deep learning is employed to detect events constituting a pattern match in the realm of CEP. To assess our approach, we performed extensive empirical testing on various scenarios with both real-world and synthetic data. We showcase examples in which our method achieves an increase in throughput of up to three orders of magnitude compared to solely employing CEP, while only suffering a minor loss in the number of detected matches.

CCS CONCEPTS

• **Information systems → Retrieval efficiency**.

KEYWORDS

Deep learning, Neural networks, Complex event processing

ACM Reference Format:
Adar Amir, Ilya Kolchinsky, and Assaf Schuster. 2022. DLACEP: A Deep-Learning Based Framework for Approximate Complex Event Processing. In *SIGMOD '22, June 12–17, 2022, Philadelphia, USA.* ACM, New York, NY, USA, 16 pages. https://doi.org/XX.XX/XXX.XX

1 INTRODUCTION

Complex event processing (CEP) is employed to extract real-time patterns from massive amounts of data in highly practical areas such as healthcare, stock trading, and IoT analytics [9, 18, 96].

Let's look at the following example of a pattern matching problem in the financial domain:

Example (1). *A stock market application monitors stock prices in a data stream. The application must notify the user upon each occurrence of the following scenario: an arrival of a stock named A, followed by the arrival of a stock named B, followed by the arrival of a stock named C, such that the stock price of C is higher than the stock price of both A and B.*

The stream of stock prices is processed by an evaluation engine that outputs the pattern matches. A depiction of a general pattern matching system is provided in Figure 1(a).

We assume that each item in the stream contains the stock name and its current price p. The items of similar format are pre-partitioned and referred to as *primitive events*. The subsequent matches are the subsets of different events that conform to the pattern conditions. In Example(1), we are looking for arrival-ordered subsets of size 3 that contain specific event types and conform to certain numerical conditions.

The CEP engine evaluates each primitive event to decide whether it can participate in a match. Since all matches are required to be emitted, any event that is applicable to the pattern is stored for possible later use as part of a match. The stored events are composed together into subsets that may or may not eventually be extended into matches. A newly arrived event is combined with all currently stored subsets for possible extension. This method is wasteful in terms of processing time when the data contains many subsets that end up being discarded.

Non-deterministic finite automaton (NFA) is the most popular CEP evaluation mechanism. [16, 18]. Each NFA state represents a different match prefix. Each evaluated event instigates an automaton transition and possible prefix storage within one of its states. An NFA-based CEP evaluation is depicted in Figure 2, and illustrates the problem described earlier. In this example, there is only one match in the stream, consisting of the events A_1, B_1, and C_1. The majority of stored prefixes are discarded. The real-world domain of stocks contains many more complicated pattern examples, such as simultaneously monitoring fluctuations of dozens of stock prices.

The wasted processing time grows exponentially with the length and the complexity of the detected pattern. This calls for a revision in strategy when attempting to extract pattern matches from streams.

To overcome this challenge, we propose relaxing the constraint to return all possible matches and, in return, maintain resource efficiency throughout evaluation. This approach is known as approximate complex event processing (ACEP).

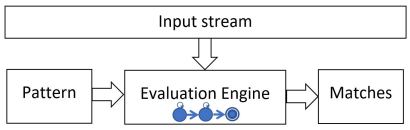
We implemented this approach in our system using deep learning methodologies. To the best of our knowledge, this is the first time a deep learning (DL) approach is being used in the field of CEP to detect primitive events constituting pattern matches within streams.

SIGMOD '22, June 12–17, 2022, Philadelphia, USA

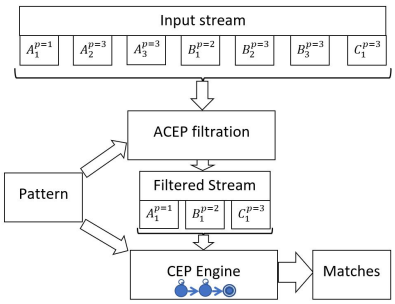
ACM ISBN 978-1-4503-XXXX-X/18/06…$15.00
https://doi.org/XX.XX/XXX.XX

Our system contains the first CEP engine that integrates neural network capabilities for complex pattern detection.

In recent years, DL models have demonstrated remarkable achievements in numerous fields [33, 47, 55, 57, 62]. The DL models designed for pattern detection have a constant processing time when it comes to most pattern or data properties, such as the amount of applicable events. This paves the way for a paradigm that extracts matches without maintaining every possible viable subset, thus significantly alleviating computational complexity.



(a) A general pattern matching system. An evaluation engine outputs pattern matches residing in the input stream.



(b) A filtration-based ACEP system integrating a CEP engine. In our system, ACEP filtration is performed by a DL model.

Figure 1: Traditional pattern matching vs. filter-based ACEP

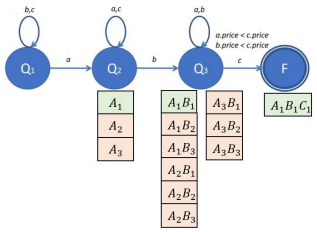


Figure 2: Example of CEP pattern matching. The input stream is the same as in Figure 1(b). Each match prefix is stored within the relevant automaton state. Orange rectangles denote discarded prefixes.

However, utilizing neural networks to fully replace the CEP process is not viable. CEP engines not only detect primitive events relevant to the targeted pattern, but also group them into pattern matches. Since neural networks serve as function approximators, the latter operation of grouping the input into subsets is extremely challenging for a neural network to learn and execute [92]. In addition, deep learning models are typically used to process unstructured data such as text or images to identify proximate temporal or spatial features. In contrast, CEP patterns encompass arbitrarily convoluted temporal and spatial dependencies with events having large temporal gaps. Consequently, a trivial solution substituting a CEP mechanism with a neural network is not sufficient and more advanced approaches are needed.

Our proposed solution, that we call DLACEP, overcomes the above difficulty by combining a DL model suited for complex pattern detection with a traditional CEP engine, with the former in charge of identifying the relevant primitive events and the latter responsible for grouping them into matches. The DL model is trained on a stream labeled according to monitored patterns. Afterwards, a new stream is evaluated by the DL model in an attempt to mark all events that participate in matches. Thereafter, only marked events are relayed to a CEP engine for match extraction. An illustration of a filtration-based ACEP system is shown in Figure 1(b). This combination can dramatically decrease processing.

The contributions of this paper can be summarized as follows:

- A formal definition and study of approximate complex event processing (ACEP), which improves the processing performance of current CEP systems.
- A novel filtration-based ACEP framework utilizing deep learning to extract applicable events from the input stream and CEP to group these events into pattern matches. To the best of our knowledge, this is the first system of its kind.
- An LSTM [31] based implementation of our novel framework. We adapted our DL models to overcome inherent problems that arise when employing neural networks to perform gradual stream processing.
- An extensive experimental evaluation of our method on real-world and synthetic data, investigating the different parameters affecting its superiority over baseline CEP. We demonstrate considerable speed-ups in real-life use cases.

The remainder of this paper is organized as follows. Section 2 provides background on CEP and deep learning, and introduces the notations used throughout the paper. In Section 3, we formally define and discuss the ACEP problem. Section 4 provides a detailed overview and specification of DLACEP. We report the results of our experimental evaluation in Section 5 and conduct a thorough analysis of them. Section 6 discusses related work while Section 7 concludes the paper and addresses our future work.

2 BACKGROUND AND TERMINOLOGY

2.1 Complex Event Processing

Overview. Formally, a primitive event is defined as a tuple (N, F, t), where N is the *event type*, $F = \{F_1, F_2, ..., F_m\}$ is an attribute set of fixed size, and t is a timestamp of the event occurrence. Events arrive to a CEP system in the form of an infinite stream, possibly from multiple sources and at different frequencies.

CEP engines match between events located within the same context called a *window*. Windows are usually either count-based or time-based. A count-based window of size W includes W subsequent events, while a time-based window of the same size contains a batch of events occurring within W time units. Adjacent windows

may intersect and overlap. Figure 3 illustrates the above. We emphasize that the number of events in each count-based window of size W is fixed at W. Windows based on different semantics [2, 56] are beyond the scope of this paper and will be covered in our future work.

Pattern conditions convey the relations between the attribute values of different events. A common implicit condition defines an order of event occurrence. Patterns including such an ordering are also called *sequence patterns*. The patterns are usually specified by domain experts and expressed using event specification languages.

Let us demonstrate the definition of a stock-related pattern:

SEQ ($GOOG\ a, APPl\ b, MSFT\ c, INTC\ d, AMZN\ e$)
WHERE $(0.55 \cdot a.vol < b.vol < 1.45 \cdot c.vol) \wedge (3 \cdot e.vol < d.vol)$
WITHIN 1 minute

A match satisfying this pattern consists of five specific stock updates with a certain volume correlation, found within one minute of each other. This specification illustrates the most basic elements constituting a *CEP pattern*: (1) an expression describing the way in which primitive events should be combined by CEP event operators, which in this case uses a simple sequence operator; (2) a list of conditions under the WHERE clause; and (3) a time window size W declaring the maximal time difference between the events in a pattern match, noted here using the WITHIN clause.

In addition to the sequence operator [93], other widely used CEP operators include negation (NEG), Kleene closure (KC), disjunction (DISJ), and conjunction (CONJ) [38]. Negation is unique in the sense that it requires certain event types or conditions to *not* appear in pattern matches. Overall, DLACEP supports all popular CEP operators: SEQ, NEG, KC, DISJ, and CONJ. DLACEP currently supports count-based window type

An additional part of the pattern definition is the selection strategy, specifying how events are selected and consumed from the input stream. In this paper, we exclusively assume the skip-till-any-match strategy, which poses no restrictions on event inclusion in a match. It was shown in [3] that this policy is the most challenging to support from the performance standpoint since it involves creation of a worst-case exponential (in the number of events in a window) number of partial matches.

The requirement to emit all stream matches imposes the examination and possible storage of every primitive event. Events deemed applicable are iteratively assembled together, possibly resulting in a pattern match. A set of events that meets all pattern requirements imposed on them but lacks additional applicable events is referred to as a *partial match*. In Figure 2, all denoted proper prefixes such as A_1B_1 are partial matches. A set of primitive events that fulfill all pattern requirements including the required match length is denoted as a *full match*.

Limitations of CEP. When events arrive at a high rate, evaluation time quickly becomes a performance bottleneck since the number of partial matches is exponential to the number of events within the time window [93]. A number of research methods have been introduced to increase evaluation efficiency, as detailed in Section 6.

2.2 Deep Learning

RNNs. Recurrent neural networks (RNNs) are designed to capture temporal dependencies within sequences of data. It has been shown that over long sequences, RNNs demonstrated biases towards more recent inputs in the sequence, due to the vanishing and exploding gradients problem [59, 78].

LSTMs. To combat the above issue, long short-term memory (LSTM) network architecture was proposed [31]. Due to its ability to model long-term dependencies, it can be employed on large sequences such as data streams. This network accepts a vector $x = (x_1, x_2, .., x_n)$ as an input and processes the vector from left to right in *time-steps*. In each time step, a same-sized sequence denoted as a *hidden vector* $h = (h_1, h_2, .., h_n)$ is emitted. The hidden vector encapsulates information about the sequence at each time-step.

Further improving upon the LSTM model is the BiLSTM model [26]. A BiLSTM layer is composed of two separate LSTM layers. An input sequence is evaluated from left to right by one layer and from right to left by the other, relaying a *forward hidden vector* and a *backward hidden vector*, respectively. This allows the model to use both past and future context to perform more knowledgeable predictions. This is especially important in the realm of CEP, where an event is often determined to be part of a full match only after the examination of future stream events.

CRFs. BiLSTMs are often used in combination with conditional random fields (CRFs) [43, 79] to tackle sequence labeling prediction problems where neighboring inputs demonstrate complex dependencies. A CRF layer models a joint label distribution by capturing dependencies across adjacent labels. BI-CRF [58] is a bi-directional version of CRF that allows us to model more convoluted dependencies.

Performance bound. A network generalization bound is typically an upper bound on the test error, based on some quantity that can be calculated on the training set. For most networks, their generalization bound remains largely unexplained [4]. For this reason, we are unable to give exact performance bounds for our networks; instead, we empirically demonstrate the effectiveness of our solution on various cases in the experimental section. We refer the reader to a comprehensive study of generalization bounds [35], showcasing the effects of different network types, hyperparameters, and datasets on generalization capacity.

3 ACEP PROBLEM FORMULATION

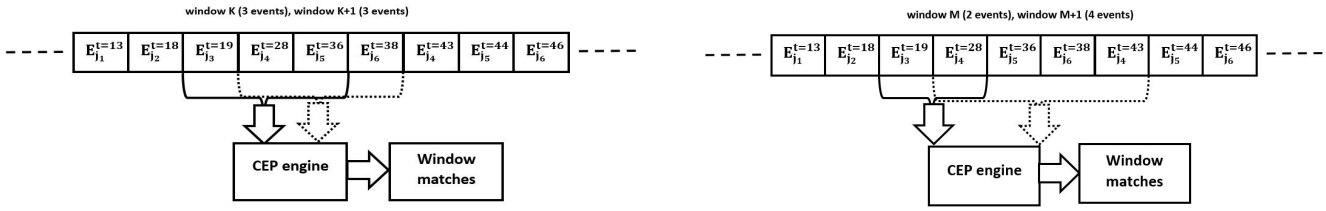
In this section, we formally define and analyze the approximate CEP (ACEP) problem that will be solved in the next section.

3.1 Formal Definition

We start by defining a global set of pattern matches within a stream.

Definition (1). Let s be a stream of primitive events $\mathbf{e_1}, \mathbf{e_2}, ..., \mathbf{e_n}$, and let $\mathcal{P}$ be a monitored set of patterns with window sizes W. We define the union set of all window matches of patterns in $\mathcal{P}$ within the stream as $M(s)_{\mathcal{P}}$.

Without loss of generality, we assume $|s|$ and therefore $|M(s)_{\mathcal{P}}|$, is finite; this can be readily extended to the infinite case. The problem that exact CEP (ECEP) solves can be formulated as follows:



(a) Count-based window evaluation.The fixed window size is 3.

(b) Time-based window evaluation. The maximal timestamp difference is 15.

Figure 3: Examples of count-based and time-based CEP evaluation methods.

Definition (2). Let s be a primitive event input stream and $\mathcal{P}$ a set of monitored CEP patterns in the stream with window sizes W. The CEP matching problem is to output the set $M(s)_{\mathcal{P}}$.

This is the conventional CEP problem addressed by CEP engines using some internal mechanism. However, this problem does not take performance metrics, such as throughput, into consideration. We denote T as a set of monitored parameters regarding the matching process. In ACEP, we aim to minimize an objective function $F_{M(s)_{\mathcal{P}},T}$, which relates to both the contents of $M(s)_{\mathcal{P}}$ and T.

Definition (3). Let s be an input stream of primitive events, $\mathcal{P}$ a set of monitored CEP patterns, and T a set of monitored parameters. The approximate CEP (ACEP) problem is to minimize $F_{M(s)_{\mathcal{P}},T}$.

An ACEP mechanism $X^{'}$ is similar to ECEP mechanisms in one major way: it outputs a set containing subsets of primitive events.

Definition (4). Let s be an input stream of primitive events and $X^{'}$ be an ACEP mechanism. While evaluating the stream s, $X^{'}$ will output a set of event subsets defined as $M(s)_{\mathcal{P}}^{X^{'}}$.

Unlike $M(s)_{\mathcal{P}}$ being output by every ECEP mechanism, each run of an ACEP mechanism $X^{'}$, or of different ACEP mechanisms, may output a different set. In addition, $M(s)_{\mathcal{P}}^{X'}$ may contain subsets of primitive events that do not constitute full pattern matches and may not contain all full pattern matches. Both $M(s)_{\mathcal{P}}^{X^{'}}$ and T are used as input to $F_{M(s)_{\mathcal{P}},T}$. The objective function we aim to minimize in ACEP can vary. An example of such an objective function is a negative weighted sum of the throughput and the emitted matches set similarity relative to some ECEP evaluation mechanism X:

$$F_{M(s)_{\mathcal{P}},T}(M(s)_{\mathcal{P}}^{X^{'}}, \{t, t^{'}\}) = -w_1 \cdot \frac{|M(s)_{\mathcal{P}} \cap M(s)_{\mathcal{P}}^{X^{'}}|}{|M(s)_{\mathcal{P}} \cup M(s)_{\mathcal{P}}^{X^{'}}|} - w_2 \cdot \frac{t}{t^{'}}.$$

where $0 \leq w_1, w_2 \leq 1, w_1 + w_2 = 1$, and $t, t^{'}$ are the throughput of $X, X^{'}$ upon the stream s, respectively.

This function does not have a minimum in $\mathbb{R}$. In practice, the value of $F_{M(s)_{\mathcal{P}},T}$ is used as a score that determines the performance of $X^{'}$ in relation to ECEP or other ACEP mechanisms. In the case of the specific $F_{M(s)_{\mathcal{P}},T}$ above, the performance metrics are throughput gain and match similarity. ACEP solutions utilizing this objective function are aimed at reducing excess processing while remaining as accurate as possible.

There are no limitations on how an ACEP mechanism $X^{'}$ attempts to achieve a minimized score. For example, it may employ a CEP mechanism X during its evaluation. It can also attempt to filter out events that do not participate in matches to improve processing performance. In our ACEP solution, we implemented a filtration-based system that filters the input stream s to output a new stream $s^{'}$, which is then conveyed to a CEP mechanism for match extraction.

3.2 Complexity Analysis

In this subsection, we analyze the computational complexity of ECEP and a filtration-based ACEP solution.

The heaviest computational burden imposed on CEP mechanisms is to create and extend partial matches to more complete partial matches or to full matches. Therefore, we calculate the computational complexity of ECEP by enumerating the number of partial and full matches within a given window. This is based on applicable event arrival rates and pattern predicate selectivity, as formulated in [39].

Let P be a monitored pattern with required event types $E_1, E_2, ..., E_n$. Let $sel_{i,j}$ be the selectivity of the predicates between E_i and E_j. The selectivity $sel_{i,j}$ shall be defined as the probability that a partial match contains events of type E_i and E_j, i.e., all pattern conditions between the two events are upheld. Let r_i be the arrival rate of event type E_i to the system. Let us denote W as the pattern window size. We can witness that the expected number of events of type E_i within a window of size W is $W \cdot r_i$. Therefore, the expected number of partial matches of all sizes (1 to $n-1$) and full matches (size n) is:

$$\Phi(W, R, SEL) = \sum_{i=1}^{n} W^i \cdot \prod_{k=1}^{i} r_i \cdot \prod_{1 \leq k,t \leq i; k \leq t} sel_{k,t}$$

Where R is the vector of all arrival rates of events with types $\{E_1, .., E_n\}$, and SEL is the vector of all predicate selectivities between events with the aforementioned types.

This means that the overall computational complexity of ECEP, which we define as the number of the partial matches of all sizes and the full matches, is simply $C^{ECEP} = \Phi(W, R, SEL)$.

Now let us consider a filtration-based ACEP system that employs match extraction on a filtered stream using CEP; ideally, the filtered

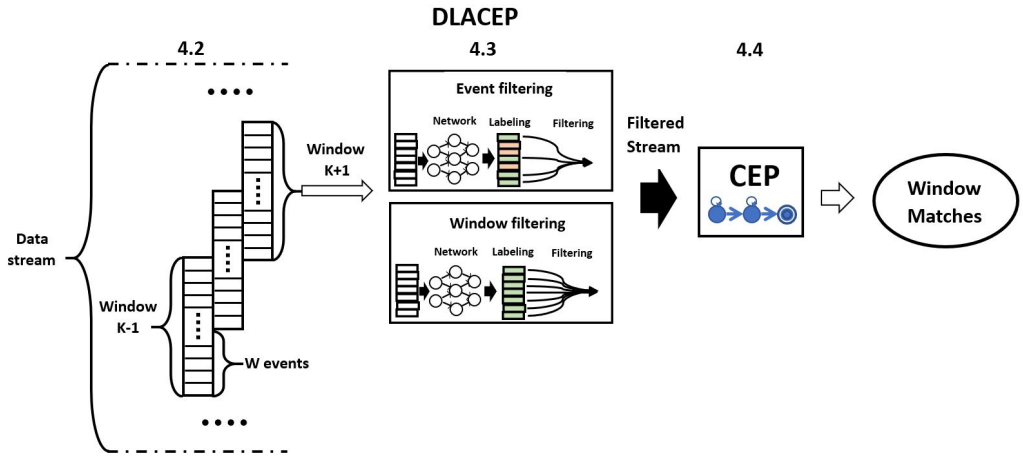


Figure 4: An overview of DLACEP. Section numbers detailing each system component are written above the relevant component. Windows of size $2 \cdot W$ are evaluated in step sizes of W events. Two event filtering schemes are shown: individual events or whole windows. A filtered stream is relayed to a CEP engine for match extraction. The final set of matches is the unification of all window matches.

stream contains only events that participate in full matches. We denote Ψ_i as the expected filtering ratio of events of type i filtered out from the stream. We also denote $R_\Psi = \langle (1-\Psi_1) * r_1, .., (1-\Psi_n) * r_n \rangle$. The computational complexity of ACEP is therefore:

$$C^{ACEP} = \underbrace{\Phi(W, R_\Psi, SEL)}_{C^{filteredcep}} + C^{filter}.$$

where C^{filter} is the computational complexity of the filtration process, and $C^{filteredcep}$ is the computational complexity of a CEP mechanism evaluating the filtered stream. We note that unlike C^{ECEP}, C^{filter} may be constant with regard to the number of partial matches in the stream, allowing it to be significantly smaller than C^{ECEP}. If C^{filter} is significantly smaller than C^{ECEP}, high filtering ratios (large $\Psi_i^{'s}$ leading to a small $C^{filteredcep}$) combined with large amounts of partial matches (large C^{ECEP}) will mean considerably higher throughput for ACEP as compared to ECEP.

If the stream contains only small amounts of partial matches, ECEP throughput may be equal to or even better than that of an ACEP solution, due to the filtering overhead. For example, let us assume a stream s contains small amounts of partial matches leading to $C^{ECEP} << C^{filter}$. Therefore, we can derive that $C^{ECEP} << C^{ACEP}$, meaning *ACEP* will perform much worse than *ECEP*. The lack of partial matches stems from short patterns or window sizes, restrictive pattern conditions, or simply a shortage of applicable events in the data. The more partial matches there are, the bigger the potential throughput gain ACEP can achieve. However, an abundance of partial matches does not guarantee a considerable increase in throughput. The amount of full matches in the data may also play a key role. Let us assume stream s contains many partial matches leading to $C^{filter} << C^{ECEP}$. However, let us also assume that $\forall i, \Psi_i = 0.001$, indicating that the vast majority of partial matches are completed to full matches. This implies that $C^{filteredcep} \cong C^{ECEP}$, which means that $C^{ACEP} \cong C^{ECEP}$. In this case, the *ACEP* mechanism will have no advantage over an *ECEP* mechanism.

A large proportion of partial matches being completed to full matches usually derives from permissive pattern conditions.

4 DLACEP

In this section we present DLACEP, our novel match extraction system. The system implements filtration-based ACEP as described in the previous section. This filtration process is carried out by dedicated deep neural networks.

System settings. We assume that the data either arrives from a single source, or is merged into one. Merging multi-source inputs in a single in-order stream is an active area of research, and is beyond the scope of this paper [46, 49, 77]. For clarity of presentation, we also assume that the system receives a single complex pattern for detection. In Section 4.3, we show how to extend our solution to the multi-pattern case. Finally, we assume the pattern window to be count-based as illustrated in Figure 3(a). In many real world domains, such as healthcare and IoT [9, 96], the sampling rate is often time constant, and therefore each time based pattern can be converted to a count based pattern. We denote the window size of the examined pattern as W. In future work, we aim to examine scenarios in which gaps between data events are not time synchronous, thus employing our system on time-based window evaluation rather than count-based. Experiments detailing simulated time-based evaluation are described in Section 5.2.

4.1 System Overview

An overview of our DLACEP solution is illustrated in Figure 4. The system is designed around a modular three-step solution. The first step features an input assembler that partitions the stream into windows of events to be processed at each evaluation step. The second step contains a trained neural network that attempts to mark the primitive events that participate in full matches within each

input window. Events are filtered out from the stream according to their issued markings, creating a filtered stream. Next, a CEP engine evaluates the filtered stream to extract all matches that exist within each filtered window. The union of all filtered window matches is the final set of matches in the data, as determined by the system.

We chose BiLSTM [26] as the architecture for our network. Two reasons have dictated this decision. First, BiLSTM networks have demonstrated groundbreaking results in the field of sequence labeling (see Section 6), and the problem of event stream filtering can be viewed as sequence labeling with 2 possible labels. Second, BiLSTM was empirically shown to be superior to other approaches such as TCN [45] and LSTM-CNN hybrid architecture in our preliminary experiments.

In essence, our DL model evaluates a stream s and outputs a filtered event stream $s^{'}$, which is then evaluated by a CEP mechanism X. The stream $s^{'}$ is filtered such that $s^{'} \subseteq s$. Our system aims to minimize the amount of matches lost and maximize the throughput as compared to employing ECEP on the unfiltered stream by X.

4.2 DNN Input Assembler

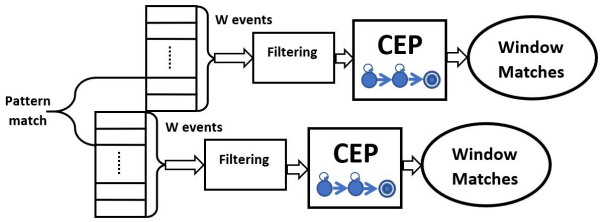


Figure 5: Example of $StepSize, MarkSize = W$ resulting in a missed pattern match.

When processing a newly seen stream, the trained LSTM type model evaluates it in fixed step sizes of W[1], marking windows of $2 \cdot W$ events each time. The evaluation step and the processing size are needed since LSTM type networks operate on sequences and not singular events. Events that are marked as participants in a complete match in each input window retain their original properties and inner order within the filtered stream.

A step size value of W and marking size of $2 \cdot W$ ensure that the network can detect all matches conforming to the original pattern window size of W. For example, let us assume the network marks only W events at each step, while retaining step sizes of W. In this case, events marked within the second half of an evaluation step and the first half of the next one will not be considered within the same context for pattern matching, even when they actually form matches together. An illustration of this scenario is presented in Figure 5.

The network could also mark W events at each processing step, advancing one event after each step. This is the way an ECEP mechanism would process the input stream. However, this marking scheme will rarely outpace ECEP, as it leads to a substantial filtering overhead.

[1]As described in Section 2, we consider fixed-sized count-based windows of W consecutive events.

Generally, we can choose to have the network mark $MarkSize$ events at each processing step and evaluate the stream in step sizes of $StepSize$. Given the pattern count window size is W, $MarkSize$ may be any positive number $MarkSize >= W$, and $StepSize$ may be any positive number $StepSize >= max\{1, MarkSize - W\}$.

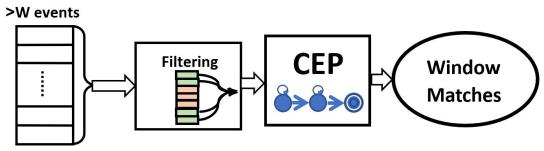


Figure 6: Example of $MarkSize > W$ resulting in detecting a pattern match participating in a count window size strictly larger than W. This leads to excess CEP processing. The CEP engine will process the marked events but will not output this match, as it does not conform to the original pattern count window size of W.

Enlarging $MarkSize$ means that more events participating in matches can be found at each processing step, thereby reducing the overall steps required. However, recall that the count window size of the original pattern is W. If $MarkSize > W$, then the network is able to detect matches contained in count-sized windows that are bigger than W, thereby not conforming to the original pattern. This may lead to excess processing. This scenario is depicted in Figure 6.

Increasing the $StepSize$ exponentially decreases the processing time since it reduces the overall amount of processed windows. However, this comes at the potential cost of overlooking matches if not all stream events are evaluated.

As explained in Section 5.1, the values of $MarkSize = 2 \cdot W$ and $StepSize = W$ were chosen as they provide a good balance between match recall and throughput gain.

$MarkSize = 2 \cdot W$ and $StepSize = W$ means that events may potentially be relayed twice for CEP evaluation. However, in practice, duplicate events are erased before being relayed.

4.3 DNN-Based Filter

The DNN-Based filter employs DL to mark events participating in a full match within each input window, and then filters out unmarked events.

Networks overview. We implemented two variations of a BiLSTM based network. The first network features several stacked BiLSTM layers and a Bi-CRF output layer (Section 2.2) that assigns a label for each event within the input window. We refer to this DL model as the *event-network* model. The second model also features several stacked BiLSTM layers. However, it attempts to detect whether an entire input window is applicable, producing a single label at its linear output layer. An input window is deemed applicable only if it contains at least one full pattern match. We refer to this model as the *window-network* model.

A detailed architectural illustration of the event-network model is given in Figure 7; the window-network model is similar in design, except for the aforementioned difference in the last layer.

Embedding and sample labeling. For each examined pattern, we use a historical data stream for network training and testing.

The stream is divided into continuous, even sized samples, each containing $2 \cdot W$ events. Each primitive event is embedded as a vector representation of pattern-relevant attributes. Categorical attributes such as the event type are represented as one-hot vectors, which can be compacted according to the pattern specifications. For example, let us assume there are 500 different event types in the data, but only one event type is specified in the pattern. The one-hot encoding can be of size 2, representing 2 categories: the applicable event type and every other type.

The training samples are binary labeled according to the examined pattern. For the event-network model, the binary labeling is issued per event in each window sample. Any event participating in a full match within the window sample gets the label 1, and any other event is labeled with 0. For the window-network model, the binary labeling is issued per window sample. Any sample that contains at least one match receives the label 1, and any other samples receive the label 0.

Since each sample size is $2 \cdot W$, the networks are essentially trained to detect patterns within *double* the window size of the original pattern.

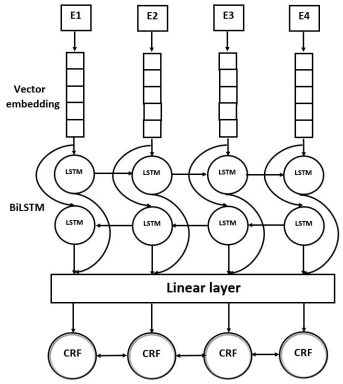


Figure 7: BiLSTM BI-CRF architecture for event labeling. Input events are vector embedded. BI-CRF outputs the most probable labels based on modeled joint label distribution. Additional information on BILSTM and CRF can be found in [26, 43, 58, 79].

Training evaluation. To evaluate the performance of the DL models during training, we calculated the F1 score on the entire test set. This score takes into account both the recall and precision of the DL model:

$$F_1 = 2 \cdot \frac{precision \cdot recall}{precision + recall}$$

The precision of the DL model can be interpreted as the probability that an entity marked as applicable by the model is truly relevant (e.g., for the event-network model, the probability that an event marked as a full match participant is indeed part of a full match). The recall of the DL model is the probability that a relevant entity is indeed marked as such by the model [25]. The entities are either events in the case of the event-network model, or windows in case of the window-network model.

Low network precision incurs an excess of events to be evaluated by the system's CEP mechanism; this reduces the overall throughput without improving accuracy. Low recall implies a scarcity in correctly marked events, reducing match accuracy.

Therefore, during training, we strive for as high an F1 score as possible. A high score serves as an indicator that the system should achieve high throughput and an ample amount of emitted matches when evaluating newly seen streams.

When there is more than one monitored pattern, we can train the network with samples labeled according to the monitoring requirement, thus semantically unifying the patterns into one. For example, given the patterns $\mathcal{P}_1$ and $\mathcal{P}_2$, let us assume we are required to monitor the matches of both patterns. Window samples are labeled with 1 if they contain at least one full match of *either* pattern. Similarly, events in each window sample are labeled with 1 if they participate in a full match of either pattern.

Filtration complexity. Recall the complexity analysis from Section 3.2. BiLSTM processing time complexity is known to be $O(h \cdot l)$, where h is the number of network parameters and l is the input sequence size [69]. Consequently, when employing BiLSTM, the filtering overhead is $C^{filter} = O(h \cdot l)$. This number is linearly dependent on internal parameters such as the number of network layers and the event embedding size. Unlike the computational complexity of ECEP, there is no dependence on the number of partial or full matches within a given window. In addition, BiLSTM filtering overhead is only linearly dependent on the window size W, whereas ECEP complexity is exponentially dependent on W. This allows for considerable throughput gains in different scenarios. In terms of the network's generalization bounds, we once again refer the reader to a comprehensive study performed in this area [35].

For each new pattern, we must retrain the model from scratch due to different labeling distribution in the training data. Training a network to full convergence can take a considerable amount of time, which is problematic in practical applications where the monitored pattern can frequently change. We worked on mitigating the retraining effects in the experimental section. In future work, we aim to thoroughly address scenarios in which the stream used during the training phase no longer captures the statistical characteristics of the most recent stream inputs, a phenomenon known as concept drift. There are two major strategies for addressing this issue:

1) Model retraining. A straightforward way of handling concept drifts is to retrain our model on a periodic basis. While reliably mitigating significant concept drifts, this approach could be highly inefficient in terms of the training time overhead. In addition, periodical model retraining requires extra memory for storing up-to-date stream samples to be later used during the retraining procedure. Recent work has attempted to expedite retraining of machine learning models [88]. Another possible avenue for reducing the training overhead is to employ transfer learning methods [80] when multiple patterns with only slight differences are detected or the changes in the training data are minor.

2) Online learning [32] involves incremental retraining by passing data instances sequentially, making it possible for the network to continuously adapt to the input stream. However, most of the prominent deep learning architectures today can only learn in an offline, batch setting, while deep online learning is considered an

open challenge [68]. Specifically, our BiLSTM model cannot be used out-of-the-box in an adaptive online learning setting and further research is required [83] to support online learning in DLACEP.

4.4 CEP Extractor

Once created, the filtered stream is evaluated by a CEP engine to extract any underlying full matches. We note that marked events relayed for match extraction after filtration do not retain their original adjacency relations to other events. Executing count-based evaluation of size W on the filtered stream can therefore procure matches that are not emitted by an ECEP mechanism evaluating the original stream.

As a result, we formulated a method in which we attach a unique increasing ID tag to each primitive event upon its arrival to the system. We then configure the system's CEP mechanism to output only sets of matches in which the events' ID values do not differ by more than $W - 1$. This ensures that our system cannot emit false positive matches. In other words, the set of matches $M(s)'_{\mathcal{P}}$ emitted by our system is fully contained in the original set of matches $M(s)_{\mathcal{P}}$ emitted by an ECEP mechanism. For example, real-time security systems in which each positive event indicates a breach can benefit greatly from eliminating false positives, as handling breach alerts require the allocation of numerous resources [7].

The exception to this rule is when our system operates to extract matches of a pattern containing a negation operator. For example, let us examine the pattern SEQ(A,B,NEG(C),D,E). Let us assume the DLACEP inner neural network erroneously marks specific events of type A,B,D,E within a certain window even though they do not participate in a full match, and does not mark any events of type C. In this case, the DLACEP inner CEP mechanism will evaluate these events in the filtered stream and incorrectly integrate them to a full match, as there is no event of type C in between. When we first examined our event-network system on negation patterns, we witnessed a large amount of false positive matches leading to poor results. We therefore altered our system in the case of negation operators, such that the event-network aims to not only label events participating in a pattern match, but also events residing under a negation operator. This led to a dramatic decrease in false positive matches in examined cases, as demonstrated in Section 5.2.

5 EXPERIMENTAL EVALUATION

5.1 Experimental Setup

Implementation details. We implemented the window-network and the event-network systems described in Section 4. Unless stated otherwise, the DL models contain 3 stacked BiLSTM layers, with each layer having a hidden vector dimension of 75. For network training, we used a varying training batch size and a dynamic learning rate [67]. DLACEP, as well as the ECEP baselines, were implemented over the OpenCEP framework [36], incorporating state-of-the-art algorithms from [37, 40, 41]. The batch size varied from 512 to 256, while the learning rate changed from 0.001 to 0.0001. All networks were trained until convergence, defined as the first epoch in which the loss remained within a 0.01 threshold for over 5 consecutive epochs The DL models were implemented in Keras [13] and trained on a GeForce RTX 2080 Ti GPU.

The event-network is based on BiCRF [58], while the window-network is a standard BiLSTM network with an output classification layer. For the event-network, the employed loss function was one that maximizes the likelihood probability sums of correct sequences in the training set for both forward and backward CRF layers [58]. We used standard binary cross-entropy loss for the window-network.

We performed preliminary experiments to discover values of $MarkSize$ and $StepSize$ that achieve a good balance between match recall and throughput gain. This resulted in $MarkSize = 2 \cdot W$ and $StepSize = W$, which were the values used for the rest of our empirical study.

All experiments were run on a machine with 80 cores having 2.10 GHz CPU and 377.0 GB RAM; a single core was used for ECEP processing and network inference. To ensure a fair comparison, all ECEP and ACEP algorithms were executed in single-threaded mode. The experiments took a total of over three months to complete.

Datasets. We used a real-world dataset purchased from the NASDAQ stock market historical records [1]. The dataset contains a total of $689,459,712$ events spanning a period of 4 years and 10 months, comprising over 2500 different stock identifiers. Each event contains a stock identifier, a timestamp, and 5 additional numerical attribute values. To preprocess the data, we re-ordered it based on the event's timestamp, divided it into $2,298,199$ chunks of 300 event-sized window samples, and removed all numerical attributes except for the stock volume attribute. The latter was standardized and a new ID was devised for each event in the data, as explained in Section 4.3.

We also employed a number of synthetic datasets, where each event contained a type and a numerical attribute. Event types were uniformly sampled from 15 possibilities. The attribute values were sampled from a standard normal distribution.

The window samples were divided at random into 70% for the train set and 30% for the test set; each sample was both window-labeled and event-labeled, as explained in Section 4.3.

Pattern queries. Tables 1 and 2 list the pattern templates used for the real-world and synthetic datasets, respectively. From each template, multiple patterns were instantiated by assigning values to the template arguments. The chosen patterns cover a wide range of values for important parameters that we describe below.

In all patterns, unless stated otherwise, the window size is $W = 150$. This value was chosen because it is both high enough to allow for substantial throughput gains and low enough to maintain exceptional system recall values and sufficient amounts of window samples. We explored the use of different values of W in a dedicated experiment (Figure 13(a)-13(b)).

Performance metrics. To compare our system's performance with that of ECEP, we used a fixed amount of consecutive events from test sets acting as the input stream. This amount varied from 20K to 40K window samples. The ECEP system evaluated the stream using a count-based evaluation of size W. Both the ECEP mechanism and our DLACEP system acted to extract the same pattern matches from the stream. To determine the merit of our system, we then measured both the throughput gain of our system over ECEP and the two sets of matches returned. We define throughput as the number of events processed per second. The quality of the returned

Q_1^A	**SEQ** $(S_1,S_2,...,S_j)$ **WHERE** $\forall t \in [j]$: $S_t \in T_k$, $p \in P([j-1])$, $\beta > \alpha > 0$ **AND** $\forall i \in p : \alpha \cdot S_i.vol < S_j.vol < \beta \cdot S_i.vol$
Q_2^A	**SEQ** (S_1,S_2,S_3,S_4,S_5) **WHERE** $\forall t \in [5]$: $S_t \in T_k$
Q_3^A	**SEQ** $(S_1,S_2,...,S_j)$ **WHERE** $\forall t \in [j]$: $S_t \in T_k$, $r \in [j]$, $p \in P([r-1])$, $l, m \in [j]$ $\beta > \alpha > 0, \gamma > 0$ **AND** $\forall i \in p : \alpha \cdot S_i.vol < S_r.vol < \beta \cdot S_i.vol, \gamma \cdot S_l.vol < S_m.vol$
Q_4^A	**SEQ** $(S_1,S_2,...,S_j)$ **WHERE** $\forall t \in [j]$: $S_t \in T_k$, $p \in P([j-1])$, $l, m \in [j]$ $\beta > \alpha > 0, \delta > \gamma > 0$ **AND** $\forall i \in p : \alpha \cdot S_i.vol < S_j.vol < \beta \cdot S_i.vol, \gamma \cdot S_l.vol < S_m.vol < \delta \cdot S_l.vol$
Q_5^A	**SEQ** $(S_1,...,S_5,$**KC**$(S_1^{'}),...,$**KC**$(S_j^{'}))$ **WHERE** $\forall t \in [5]$: $S_t \in T_{100}$, $\forall l \in [j]$: $S_l^{'} \in T_{100+l\cdot 10}/T_{100+(l-1)\cdot 10}$, $\beta > \alpha > 0$ **AND** $\forall i \in [5] : \alpha \cdot S_i.vol < S_5.vol < \beta \cdot S_i.vol$
Q_6^A	**KC(SEQ** $(S_1,S_2,...,S_j)$**)** **WHERE** $\forall t \in [j]$: $S_t \in T_{100}$, $\beta > \alpha > 0$ **AND** $\forall i \in [j-1] : \alpha \cdot S_i.vol < S_j.vol < \beta \cdot S_i.vol$
Q_7^A	**SEQ** $(S_1,...,$**NEG**$(S_1^{'}),...,$**NEG**$(S_j^{'}),S_5)$ **WHERE** $\forall t \in [5]$: $S_t \in T_{100}$, $\forall l \in [j]$: $S_l^{'} \in T_{100+l\cdot 10}/T_{100+(l-1)\cdot 10}$, $\beta > \alpha > 0$ **AND** $\forall i \in [5] : \alpha \cdot S_i.vol < S_5.vol < \beta \cdot S_i.vol$
Q_8^A	**SEQ** $(S_1,...,$**NEG(SEQ**$(S_1^{'},...,S_j^{'})$**)**$,S_5)$ **WHERE** $\forall t \in [5]$: $S_t \in T_{100}$, $\forall l \in [j]$: $S_l^{'} \in T_{100+l\cdot 10}/T_{100+(l-1)\cdot 10}$, $\beta > \alpha > 0$ **AND** $\forall i \in [5] : \alpha \cdot S_i.vol < S_5.vol < \beta \cdot S_i.vol$
Q_9^A	**DISJ(**$SEQ_1(S_1,S_2,...,S_j)$,$SEQ_2(S_1^{'},S_2^{'},...,S_j^{'})$**)** **WHERE** $\forall t \in [j]$: $S_t \in T_{100}$, $\forall l \in [j]$: $S_l^{'} \in T_{200}/T_{100}$, $\beta > \alpha > 0, \delta > \gamma > 0$ **AND** $\forall i \in [j-1] : \alpha \cdot S_i.vol < S_j.vol < \beta \cdot S_i.vol, \gamma \cdot S_i^{'}.vol < S_j^{'}.vol < \delta \cdot S_i^{'}.vol$
Q_{10}^A	**DISJ(**$SEQ_1(S_1^1,S_2^1,S_3^1,S_4^1),...,SEQ_j(S_1^j,S_2^j,S_3^j,S_4^j)$**)** **WHERE** $\forall l \in [j], \forall m \in [4]$: $S_m^l \in T_{100+(l-1)\cdot 100}/T_{100+(l-2)\cdot 100}$, $\forall r \in [j], \alpha_2^r > \alpha_1^r > 0$ **AND** $\forall i \in [j], \forall p \in [3] : \alpha_1^i \cdot S_p^i.vol < S_4^i.vol < \alpha_2^i \cdot S_p^i.vol$
Q_{11}^A	**CONJ/SEQ** $(S_1,S_2,...,S_5)$ **WHERE** $\forall t \in [5]$: $S_t \in T_{40\cdot t}/T_{40\cdot(t-1)}$, $\beta > \alpha > 0$ **AND** $\forall i \in [4] : \alpha \cdot S_i.vol < S_5.vol < \beta \cdot S_i.vol$
Q_{12}^A	**DISJ(**$SEQ_1(S_1,S_2,...,S_5)$,$SEQ_2(S_1^{'},S_2^{'},...,S_5^{'})$**)** **WHERE** $\forall t \in [j]$: $S_t, S_t^{'} \in T_{40\cdot t}/T_{40\cdot(t-1)}$, $\beta > \alpha > 0, \delta > \gamma > 0$ **AND** $\forall i \in [j-1] : \alpha \cdot S_i.vol < S_5.vol < \beta \cdot S_i.vol, \gamma \cdot S_i^{'}.vol < S_5^{'}.vol < \delta \cdot S_i^{'}.vol$

Table 1: Real-world query templates used for the stock dataset experiments. T_k is the set of the top k most prevalent stock identifiers in the dataset. $P(S)$ is the power set of S. KC is Kleene closure. NEG is negation. In all patterns besides Q_7^A larger values of j, k, m, r dictates more partial matches. In Q_7^A, larger values of j dictates less full matches. In all patterns, larger values of $-|p|, \beta - \alpha, -\gamma$ or $\delta - \gamma, \alpha_2^r - \alpha_1^r$ dictate more full matches.

Q_1^B	**SEQ** (A, B, C, D, E, F) **WHERE** $\forall X \in \{C, D\}$: $0.85 \cdot X.vol < F.vol < 1.15 \cdot X.vol$, $\forall X \in \{A, D\}$: $0.85 \cdot X.vol < E.vol < 1.15 \cdot X.vol$, $0.4 \cdot C.vol < F.vol$	Largest amount of partial matches. Low amount of partial matches completed to full matches.
Q_2^B	**SEQ** (A, B, C, D, E) **WHERE** $\forall X \in \{A, B\}$: $0.85 \cdot X.vol < D.vol < 1.15 \cdot X.vol$, $\forall X \in \{B, C\}$: $0.85 \cdot X.vol < E.vol < 1.15 \cdot X.vol$	Very large amount of partial matches. Low amount of partial matches completed to full matches.
Q_3^B	**SEQ** (A, B, C, D) **WHERE** $\forall X \in \{A, B, C\}$: $0.85 \cdot X.vol < D.vol < 1.15 \cdot X.vol$	Large amount of partial matches. Low amount of partial matches completed to full matches.

Table 2: Query templates used for experiments with the synthetic datasets.

matches was evaluated using F1 (Section 4.3) for the negation patterns, where false positives are possible as explained in Section 4.4, and recall for the rest of the patterns.

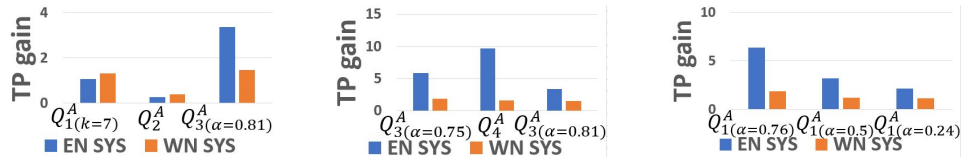
5.2 Experimental Results

There are several factors that affect the performance of DLACEP. As explained in Section 3.2, the amount of full and partial matches has a significant impact on a filter-based ACEP solution. In addition, the amount of training data, the amount of training epochs, and the network architecture have a notable impact on a DL-based ACEP solution. Pattern complexity and pattern operators can also have an impact on the neural network's performance and the CEP running time. Overall, we tested 45 different pattern instantiations demonstrating widely different performance-affecting parameters.

Our experiments exposed the pattern size and the window length (W) as the most important parameters affecting our system's performance. As explained in previous sections, the effect of these parameters on the scalability of ECEP mechanisms is paramount, while having a considerably lower impact on the processing time of neural networks.

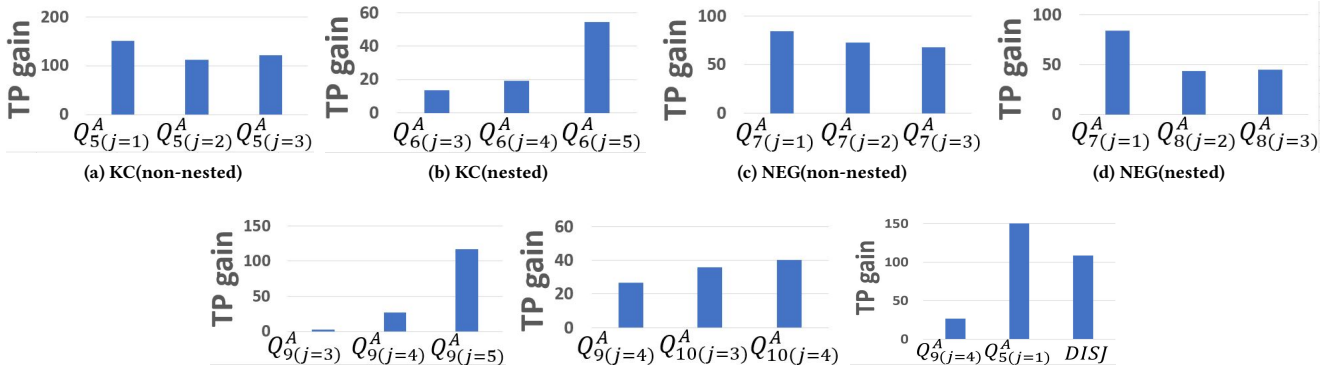
Unless stated otherwise, in all experiments, we attained very high recall/F1 scores (0.95 and above), where the window-network system achieved slightly better recall than the event network system. For each experiment, the exact pattern and parameters used, as well as the raw results (exact training times, exact matches missed, etc.) are provided in the supplementary material.

Impact of the amount of partial matches. In this set of experiments, we tested both the event-network and the window-network system on 3 patterns, displaying different amounts of partial matches. The results are displayed in Figure 8(a). As explained in Section 3.2, the amount of partial matches can signify the potential throughput gain of an ACEP system over ECEP. The pattern $Q^A_{1_{(k=7,...)}}$ demonstrates a low number of partial matches in the data. This led to a fast ECEP evaluation process, resulting in only small gains in system throughput values. The pattern Q_2^A demonstrates a high number of partial matches, where almost all of them are completed to full matches. This led to ECEP and our system's CEP mechanism having almost similar processing times. The additional filtering overhead resulted in our system having worse throughput than ECEP. The pattern Q_3^A displays large amounts of partial matches within the data with only a small fraction extended



(a) Different amount of partial matches (b) Different ratio of partial to full matches (c) Different amount of full matches

Figure 8: Impact of the amount of partial and full matches on the throughput gain over ECEP (higher is better), on 8 patterns displaying varying amounts of full and partial matches. A large amount of partial matches and a low fraction of partial matches that are extended to full matches greatly increases our system's potential for throughput gain.



(a) KC(non-nested) (b) KC(nested) (c) NEG(non-nested) (d) NEG(nested)

(e) DISJ(2 SEQ) (f) DISJ(SEQ LEN 4) (g) Separate vs. DISJ

Figure 9: Impact of the pattern operator on the throughput gain over ECEP (higher is better), on 17 patterns displaying varying operators, lengths, and nesting. DLACEP can efficiently handle patterns including all popular pattern operators.

to full matches. This led to both the window-network system and the event-network system achieving significant throughput gains over ECEP due to the large number of filtered events.

To further test the scalability of our event network system with regard to partial matches, we generated a pattern with massive amounts of partial matches where relatively few are completed to full match. The pattern is $Q^A_{1_{(k=100,\ldots)}}$. The event-network system achieved a throughput gain of 294. This empirically demonstrates the exceptional benefit of employing our system in the scenario where there are vast quantities of partial matches.

In patterns demonstrating partial match scarcity ($Q^A_{1_{(k=7,..)}}, Q^A_2$) the window-network system outpaced the event-network system due to its reduced processing times. In the case of $Q^A_{1_{(k=7,\ldots)}}$, the event network reached a recall of 0.856. The reason for this is explained in the next set of experiments.

Impact of the amount of full matches. In the next set of experiments, we tested both the event-network system and window-network system on 6 patterns displaying different amounts of full matches. The results are shown in Figure $8(b), 8(c)$. As explained in Section 3.2, the amount of full matches influences the filtering ratio value and therefore impacts an ACEP solution's throughput gain. In addition, a low amount of full matches in the data can lead to reduced match recall. This is because a deficiency of matches impairs the networks ability to properly learn the pattern concept. The latter issue is illustrated by the previously examined patterns $Q^A_{1_{(k=7,\ldots)}}$ and $Q^A_{1_{(k=100,\ldots)}}$, similar in their complexity. Many more full matches were observed for the latter pattern than for the former, leading to a sizable recall gap of 0.143 when evaluating the former pattern.

$Q^A_{3_{(..,\alpha=0.75,..)}}$, $Q^A_{3_{(..,\alpha=0.81,..)}}$, and Q^A_4 demonstrated varying amounts of full matches and partial matches in the data. Overall, Q^A_4 displayed the largest event network system throughput gain because it had the smallest fraction of partial matches completed to full matches. This increased the filtering ratio, elevating the overall throughput gain over ECEP. To better demonstrate the unique effect of the amount of full matches, we tested a number of different patterns generated from the pattern Q^A_1. All the template variations displayed the same amount of partial matches in the data, but a different amount of full matches. $Q^A_{1_{(..,\alpha=0.24,..)}}$ demonstrated the highest amount and $Q^A_{1_{(..,\alpha=0.76,..)}}$ the lowest. Consequentially, $Q^A_{1_{(..,\alpha=0.76,..)}}$ displayed the best throughput gain by a large margin. This is because, given a similar number of partial matches, the filtering ratio increases as the number of full matches decreases, alleviating the ACEP computational complexity. We note that the event-network system outpaced the window-network system in all the patterns examined because it filtered out considerably more events.

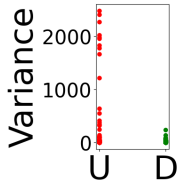


Figure 10: Distribution of the volume attribute variance value in the matches of the pattern $Q^A_{10(j=4)}$ detected (D) and undetected (U) by DLACEP.

Impact of the pattern operator. We tested the event-network system on 16 patterns containing a variety of the CEP operators listed in Section 2.1. The results are displayed in Figure 9. For disjunction, we evaluated patterns with 2 sequences of varying length (Q^A_9), and patterns with identical sequence lengths but different numbers of sequences (Q^A_{10}). For Kleene closure, we evaluated patterns with different amounts of KC operators (Q^A_5), and nested sequences (Q^A_6). For negation, we evaluated patterns with different amounts of NEG operators (Q^A_7), and nested sequences (Q^A_8). All examined patterns displayed large amounts of partial matches in the data, with a small fraction of them completed to full matches. For $Q^A_6(j = 5)$, the recall attained was 0.877, due to the increased pattern complexity. Increasing the number of nested sequences under disjunction or their length, and increasing the length of nested sequences under Kleene closure, increases the throughput gain due to the increased amount of partial matches. However, increasing the amount of negation or KC operators and the length of nested sequences under negation decreases the throughput gain. In both cases, this is due to an increase in the amount of full matches, which lead to a reduced filtering ratio. The results show that DLACEP can effectively handle all popular pattern operators.

Separate vs combined pattern evaluation.The disjunction (DISJ) operator allows to combine multiple patterns into a composite pattern returning a union of all matches. In our next experiment, we assessed the throughput gain and recall of a disjunction of two simple patterns, $Q^A_{9(j=4)}$ and $Q^A_{5(j=1)}$, as opposed to evaluating each of them separately. To simplify result interpretation, patterns with highly varying throughput gain by DLACEP were chosen. DLACEP achieved a throughput gain of 108.65 on the disjunction pattern (Figure 9(g)), which is higher than the average of the results obtained on the individual patterns. The recall for the disjunction of the two patterns was 0.997, a result identical to the higher of the two patterns when evaluated separately. Overall, we can conclude that in certain cases DLACEP can achieve better results when individual patterns are jointly defined as a disjunction. Our future research will provide an in-depth study of this phenomenon.

Qualitative analysis of matches. CEP engines are often utilized in applications that do not tolerate missing certain matches. Therefore, it is imperative to qualitatively analyze the matches missed by DLACEP. In a case study performed on the pattern $Q^A_{10(j=4)}$, we partitioned the detected (595397 in total) and undetected (5423 in total) matches separately according to the stock volume attribute value. This attribute was chosen due to its direct appearance in the pattern conditions. The results are displayed in Figure 10. It can be observed that the volume of the missed matches exhibits significant variance as compared to the detected ones. This matches our expectations as smoother volume transitions between events are easier for the network to categorize correctly. We intend to repeat this experiment on a variety of patterns and to devise a way that enables the network to better handle high variance matches in our future work.

Impact of the amount of training data and epochs. As stated in Section 4.3, for each new pattern, the DLACEP neural network needs to retrain from scratch, which may be problematic in practical applications. In addition, accumulating training data can take a considerable amount of time. Therefore, we examined the DLACEP performance after training its inner neural network for different amount of epochs and using different percentages of training data. We evaluated our event network system on the previously examined pattern $Q^A_{9_{(j=5,..)}}$. In these experiments, we evaluated the system's percentage of false negatives (FN%), which indicates the percentage of missed matches out of the entire set of matches. The results are displayed in Figure 11. We chose the pattern $Q^A_{9_{(j=5,..)}}$ because the neural network required a significant amount of training epochs (58) to reach convergence. In the data percentage experiments, the network was trained for 30 epochs, and the data was chosen randomly each time from the entire training set. For the epoch number experiments, the entire training set was used. The results show that DLACEP can efficiently produce accurate results with small amounts of training epochs and data; the FN% ratio remains low and the throughput gain remains high. The throughput gain decreases and stabilizes with increasing amounts of data and training epochs. This is due to class imbalance in favor of 0 labeled events in the data, which leads to overfiltering events at low amounts of data and epochs. The FN% stabilizes quickly, which indicates that DLACEP can achieve results similar to those achieved at full convergence, even after significantly reducing the amount of data and training epochs.

Comparison to ECEP optimizations. We compared DLACEP employing the event network to two ECEP optimization baselines that were developed to reduce ECEP processing times: ZSTREAM[54] and lazy evaluation [41]. These optimizations act on top of an ECEP mechanism to improve its processing time in a wide array of cases and have reached SOTA results. ZStream presents a dynamic programming algorithm for tree-based plan generation, utilizing a cost model based on CPU access. Lazy evaluation is a principle used to process events in an order different from their arrival order, usually by their frequency order, from highest to lowest. This often results in having to store fewer partial matches during evaluation.

We compared DLACEP with the optimization baselines on patterns containing varied pattern operators. The results are displayed in Figure 12. DLACEP achieved recall scores of 0.94279, 0.94523 and 0.81424 on the patterns $Q^A_{11(CONJ,..)}$, Q^A_{12}, and $Q^A_{11(SEQ,..)}$, respectively. The results show that DLACEP far outpaces SOTA ECEP optimizations, with only a slight loss of matches. As evident in the graphs, the optimization methods alleviate computation time and lessen the number of partial matches, but only mildly in comparison to DLACEP. Our system leads to far fewer partial matches during evaluation while incurring only a small computation overhead.

(a) Epochs to TP (b) Epochs to FN% (c) Data% to TP (d) Data% to FN%

Figure 11: Impact of data% and training epochs on the throughput gain over ECEP (higher is better) and FN% (lower is better), by evaluating the pattern $Q_9^A(j{=}5,..)$. DLACEP can efficiently handle reduced amounts of data and training epochs.

(a) $Q_{11_{(SEQ,..)}}^A$ Sequence (b) $Q_{11_{(CONJ,..)}}^A$ Conjunction (c) Q_{12}^A Disjunction

Figure 12: Comparing throughput gain over ECEP (higher is better) of ACEP with ECEP optimization algorithms, by evaluating 3 patterns. DLACEP far outpaces ECEP optimization algorithms.

Impact of the window and pattern size. We tested the event network system on 3 pattern templates displaying different pattern lengths of 4, 5, 6. We tested these patterns with different values of W, ranging from 100 to 350 in gaps of 50. The results are displayed in Figure 13(a), 13(b). For each pattern length and value of W, we generated a new synthetic dataset, where event numerical attributes were generated from the standard normal distribution. This is to ensure a fair comparison between each pair of (W, pattern length). Each pattern demonstrated significant amounts of partial matches in the data, with only a few forming full matches.

We observed considerable throughput gain for all the patterns examined and all values of W. In the case of pattern length 6 and window size 350, our system achieved a gain of 3 orders of magnitude compared to ECEP. The results showcase the vast impact of both window size and pattern length on our system's throughput and clearly demonstrate CEP scalability issues and the ability of DLACEP to cope with these. We expect our system to have an extended advantage with even bigger values of W and pattern length. However, as expected, the increased pattern complexity leads to a recall gap between the different patterns and values of W.

Impact of the amount of network layers. To address possible recall deterioration when evaluating overly complex patterns, we assessed the impact of the number of stacked BiLSTM layers in our system's network. We compared networks with 3, 4, and 5 layers activated on the pattern Q_1^B with $W = 350$. The results are displayed in Figure 13(c), (d). The throughput gain deteriorates as the number of layers increases due to the additional processing time imposed by deeper DL models. However, the recall grows with the number of layers due to higher network capacity. These experiments validate that even when the pattern complexity increases, our network is flexible enough to cope with them, given architecture or parameter tuning. However, this can impact throughput gain.

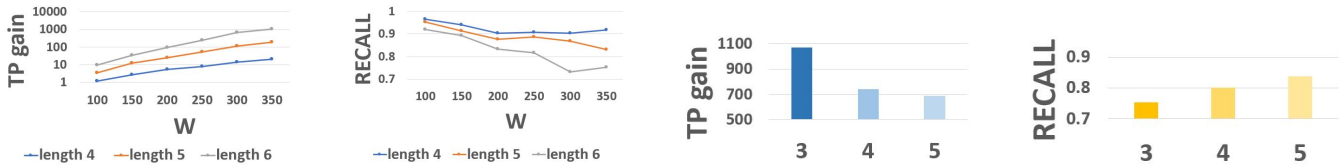
Time-based window evaluation. Unlike count-based windows, time-based windows can contain different amounts of events. As training an LSTM network requires sequences of fixed size, patterns utilizing time-based windows could negatively affect the performance of DLACEP.

To assess the extent of this issue, we simulated time-based windows by partitioning the stocks dataset into windows of randomly chosen sizes of up to MW events. During the training phase, all windows were padded to the maximal size by adding blank events. The pattern ${Q_5}_{j=2}^A$ was chosen for this experiment due to the high sensitivity of Kleene closure patterns to window size fluctuations.

The results are displayed in Figure 14. For comparison, the throughput gain achieved for the same pattern on a count-based window of size 300 can be found in Figure 9(a). While DLACEP throughput in the time-based scenario was about a half of the one measued in the count-based case, it still led to an improvement by the factor of 50. For MW values of 250 and 350 the improvement was even greater, which could be explained either by the smaller overhead of DLACEP due to a smaller time window ($MW = 250$) or by a larger number of total events as opposed to the number of relevant (unfiltered) events ($MW = 350$). The measured recall exceeded 0.95 in all tested cases. To summarize, the above results indicate that DLACEP can provide significant performance improvements also when applied on patterns with time-based window semantics.

Summary and conclusions. Our experiments show that DLACEP provides a significant performance advantage as compared to an ECEP solution in a variety of scenarios. Our system achieves higher throughput gains for increasing window and pattern lengths, exceeding those achieved by SOTA algorithms, with only a minor loss of matches.

Event and window network comparison. In cases where both systems were examined, the window-network demonstrated a slightly higher quality of returned matches. However, its coarser granularity often led to a lower filtering ratio, decreasing the overall throughput. For patterns with few partial matches, such as in Figure 8(a), the window-network system also achieved higher throughput



(a) TP to W (b) Recall to W (c) Tp gain to layers (d) Recall to layers

Figure 13: Impact of the pattern length, window size, and number of layers on the throughput gain over ECEP (higher is better) and recall (higher is better). The DLACEP efficiency increases exponentially with pattern and window size. Recall achieved on convoluted patterns increases with the number of network layers, at a cost of throughput deterioration.

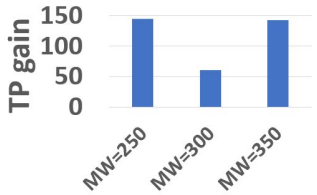


Figure 14: Impact of the max window (MW) size on DLACEP throughput gain over ECEP (higher is better) on the pattern $Q5^{A}_{j=2}$ in simulated time-based window evaluation.

due to its reduced network complexity combined with a high CEP throughput derived from partial match scarcity.

Network training. In cases where both systems were examined, training the window-network was up to five times faster than the event-network, due to the reduced network complexity and the computationally simple loss function of the former. This establishes the window-network system as more flexible and adaptable to changing patterns, while the pretrained event-network system is faster on newly seen data. Increased pattern complexity, number of BiLSTM layers and the amount of training data all demand lengthier training times when the network is trained to convergence. The actual training times ranged from 9.5 hours to 12 days, with the average training time being about 3 days.

6 RELATED WORK

CEP systems and optimizations. CEP has become an increasingly popular research field in recent years [15, 17, 21, 28, 81]. It originally derived from data management systems such as Stream [5]. Later, more expressive frameworks, such as CEDR [6], SASE/SASE+ [27, 87], and T-REX [16], allowed for richer pattern queries. The most widespread CEP mechanisms are NFAs and trees [16, 18, 54].

A plethora of methods have been developed in an attempt to optimize CEP throughput [3, 16, 19, 30, 37, 40, 41, 54, 61, 64, 87, 94]. Given a specific sequence pattern, ZStream [54] attempts to locate an optimal tree structure for its evaluation, based on a CPU cost model. Another proposed improvement is to evaluate primitive events by rate of frequency instead of arrival order. This is done by storing events in a separate buffer and evaluating less prevalent events first [41]. These optimizations are orthogonal to our solution. Our system can employ any CEP system, such as Flink [65].

ACEP research. Little effort has been invested thus far to research approximate complex event processing (ACEP). In [48], cases in which the complex pattern is ambiguous or the data stream contains errors are researched, in addition to cases where partial matches (PM) are shed to maintain moderate resource usage. In contrast, we do not address ambiguous patterns and stream errors. Instead, we incorporate a potentially imprecise evaluation mechanism, namely a neural network, to improve the system performance.

Load shedding. Shedding either PMs or stream events is referred to as load shedding [29, 75, 76, 95]. Load shedding is introduced when a CEP system needs to maintain some latency bound under resource constraints during peak times, while minimizing result degradation. Our system, however, is meant to be used as a conceptual shift towards mitigating CEP scalability issues, and not as an emergency solution used only at peak times.

Deep learning for pattern recognition. Deep learning demonstrates remarkable results in pattern recognition tasks [23, 91]. Specifically, BILSTM-CRF is widely used for a variety of learning tasks, such as sequence prediction, data extraction and identification, sentence labeling, and classification, and has demonstrated state-of-the-art results in many of these realms [10, 12, 24, 34, 44, 51, 58, 60, 66, 82]. Previous research [10, 50, 84] has shown that stacking several BILSTM layers together to create a *stacked BILSTM* improves the performance of certain classification tasks. This is supported by various theoretical examinations demonstrating that some mapping functions are represented more efficiently by deeper DL models [8, 78].

Deep learning in CEP. Applying machine learning methodologies in the context of CEP is a relatively unexplored research field. In [52, 53, 72], ML and DL techniques were employed to automate the process of rule learning and to extract meaningful patterns from data in an attempt to replace domain experts. In [89], a real-time object detection DL model is used to process streams and extract events that are combined into primitive events for further processing by a CEP engine. In [14, 22, 85], predictive CEP systems are introduced in an attempt to foresee when future complex events transpire using ML methodologies. In contrast to all these, DLACEP is the first to incorporate deep learning into the process of complex pattern detection, augmenting the core part of a CEP engine.

7 CONCLUSION AND FUTURE WORK

In this paper, we discussed the problem of efficient approximate pattern matching. A novel ACEP solution was presented in which

a DL model marks relevant events in the stream and relays them for CEP match extraction. We implemented two BILSTM-based systems and experimentally demonstrated their effectiveness in different scenarios. Our work signifies the first step towards integrating deep learning methodologies to detect events constituting complex pattern matches. Our future research efforts will focus on the various directions outlined in Section 4.

ACKNOWLEDGMENTS

The research leading to these results was supported in part by the Israel Science Foundation (grant No.191/18), by the Israel Ministry of Science and Technology, and by the Technion Hiroshi Fujiwara Cyber Security Research Center and the Israel National Cyber Directorate.

REFERENCES

[1] [n.d.]. http://www.eoddata.com/.
[2] 2021. Overview of the Event Processing Language (EPL). https://docs.oracle.com/cd/E12839_01/apirefs.1111/e14304/overview.htm#EPLLR238
[3] Jagrati Agrawal, Yanlei Diao, Daniel Gyllstrom, and Neil Immerman. 2008. Efficient Pattern Matching over Event Streams. In *Proceedings of the 2008 ACM SIGMOD International Conference on Management of Data* (Vancouver, Canada) *(SIGMOD '08)*. Association for Computing Machinery, New York, NY, USA, 147–160. https://doi.org/10.1145/1376616.1376634
[4] Zeyuan Allen-Zhu, Yuanzhi Li, and Yingyu Liang. 2018. Learning and Generalization in Overparameterized Neural Networks, Going Beyond Two Layers. *arXiv* (Nov 2018). arXiv:1811.04918 https://arxiv.org/abs/1811.04918v6
[5] Arvind Arasu, Brian Babcock, Shivnath Babu, John Cieslewicz, Mayur Datar, Keith Ito, Rajeev Motwani, Utkarsh Srivastava, and Jennifer Widom. 2016. STREAM: The Stanford Data Stream Management System. In *Data Stream Management*. Springer, Berlin, Germany, 317–336. https://doi.org/10.1007/978-3-540-28608-0_16
[6] Roger S. Barga, Jonathan Goldstein, Mohamed Ali, and Mingsheng Hong. 2007. Consistent Streaming Through Time: A Vision for Event Stream Processing. In *In CIDR*. 363–374.
[7] Lars Baumgärtner, Christian Strack, Bastian Hoßbach, Marc Seidemann, Bernhard Seeger, and Bernd Freisleben. 2015. Complex Event Processing for Reactive Security Monitoring in Virtualized Computer Systems *(DEBS '15)*. Association for Computing Machinery, New York, NY, USA, 22–33. https://doi.org/10.1145/2675743.2771829
[8] Y. Bengio. 2009. Learning Deep Architectures for AI. *Foundations and Trends in Machine Learning* 2, 1 (Jan 2009), 1–55. https://doi.org/10.1561/2200000006
[9] Marion Blount, Maria Ebling, J. Eklund, Andrew James, Carolyn Mcgregor, Nathan Percival, Kathleen Smith, and Daby Sow. 2010. Real-Time Analysis for Intensive Care: Development and Deployment of the Artemis Analytic System. *IEEE engineering in medicine and biology magazine : the quarterly magazine of the Engineering in Medicine & Biology Society* 29, 2 (May 2010), 110–8. https://doi.org/10.1109/MEMB.2010.936454
[10] Linqin Cai, Sitong Zhou, Xun Yan, and Rongdi Yuan. 2019. A Stacked BiLSTM Neural Network Based on Coattention Mechanism for Question Answering. *Comput. Intell. Neurosci.* 2019 (Aug 2019). https://doi.org/10.1155/2019/9543490
[11] Koral Chapnik, Ilya Kolchinsky, and Assaf Schuster. September 2022. DARLING: Data-Aware Load Shedding in Complex Event Processing Systems. 48th International Conference on Very Large Data Bases (PVLDB), Sydney, Australia.
[12] Tao Chen, Ruifeng Xu, Yulan He, and Xuan Wang. 2017. Improving Sentiment Analysis via Sentence Type Classification Using BiLSTM-CRF and CNN. *Expert Syst. Appl.* 72, C (April 2017), 221–230. https://doi.org/10.1016/j.eswa.2016.10.065
[13] François Chollet. 2015. keras. https://github.com/fchollet/keras.
[14] M. Christ, J. Krumeich, and A. W. Kempa-Liehr. 2016. Integrating Predictive Analytics into Complex Event Processing by Using Conditional Density Estimations. In *2016 IEEE 20th International Enterprise Distributed Object Computing Workshop (EDOCW)*. 1–8. https://doi.org/10.1109/EDOCW.2016.7584363
[15] Gianpaolo Cugola and Alessandro Margara. 2011. Processing flows of information: from data stream to complex event processing. *ACM COMPUTING SURVEYS* (2011).
[16] Gianpaolo Cugola and Alessandro Margara. 2012. Complex Event Processing with T-REX. *J. Syst. Softw.* 85, 8 (Aug. 2012), 1709–1728. https://doi.org/10.1016/j.jss.2012.03.056
[17] Miyuru Dayarathna and Srinath Perera. 2018. Recent Advancements in Event Processing. *ACM Comput. Surv.* 51, 2, Article 33 (Feb. 2018), 36 pages. https://doi.org/10.1145/3170432
[18] Alan Demers, Johannes Gehrke, Mingsheng Hong, Mirek Riedewald, and Walker White. 2006. Towards Expressive Publish/Subscribe Systems. In *Advances in Database Technology - EDBT 2006*. Springer, Berlin, Germany, 627–644. https://doi.org/10.1007/11687238_38
[19] L. Ding, S. Chen, E. A. Rundensteiner, J. Tatemura, W. Hsiung, and K. S. Candan. 2008. Runtime Semantic Query Optimization for Event Stream Processing. In *2008 IEEE 24th International Conference on Data Engineering*. 676–685. https://doi.org/10.1109/ICDE.2008.4497476
[20] Opher Etzion and Peter Niblett. 2010. *Event Processing in Action* (1st ed.). Manning Publications Co., USA.
[21] Ioannis Flouris, Nikos Giatrakos, Antonios Deligiannakis, Minos Garofalakis, Michael Kamp, and Michael Mock. 2017. Issues in complex event processing: status and prospects in the Big Data era. *Journal of Systems and Software* 127 (May 2017), 217–236. https://doi.org/10.1016/j.jss.2016.06.011
[22] Lajos Jenő Fülöp, Árpád Beszédes, Gabriella Tóth, Hunor Demeter, László Vidács, and Lóránt Farkas. 2012. Predictive Complex Event Processing: A Conceptual Framework for Combining Complex Event Processing and Predictive Analytics. In *Proceedings of the Fifth Balkan Conference in Informatics* (Novi Sad, Serbia) *(BCI '12)*. Association for Computing Machinery, New York, NY, USA, 26–31. https://doi.org/10.1145/2371316.2371323
[23] X. Gao, J. Zhang, and Z. Wei. 2018. Deep learning for sequence pattern recognition. In *2018 IEEE 15th International Conference on Networking, Sensing and Control (ICNSC)*. 1–6. https://doi.org/10.1109/ICNSC.2018.8361281
[24] Lejun Gong, Xingxing Zhang, Tianyin Chen, and Li Zhang. 2021. Recognition of Disease Genetic Information from Unstructured Text Data Based on BiLSTM-CRF for Molecular Mechanisms. *Secur. Commun. Netw.* 2021 (Feb 2021). https://doi.org/10.1155/2021/6635027
[25] Cyril Goutte and Eric Gaussier. 2005. A Probabilistic Interpretation of Precision, Recall and F-Score, with Implication for Evaluation. In *Advances in Information Retrieval*. Springer, Berlin, Germany, 345–359. https://doi.org/10.1007/978-3-540-31865-1_25
[26] A. Graves, A. Mohamed, and G. Hinton. 2013. Speech recognition with deep recurrent neural networks. In *2013 IEEE International Conference on Acoustics, Speech and Signal Processing*. 6645–6649. https://doi.org/10.1109/ICASSP.2013.6638947
[27] Daniel Gyllstrom, Eugene Wu, Hee-Jin Chae, Y. Diao, Patrick Stahlberg, and Gordon Anderson. 2006. SASE: Complex Event Processing over Streams. *ArXiv* abs/cs/0612128 (2006).
[28] S. Hallé and S. Varvaressos. 2014. A Formalization of Complex Event Stream Processing. In *2014 IEEE 18th International Enterprise Distributed Object Computing Conference*. 2–11. https://doi.org/10.1109/EDOC.2014.12
[29] Yeye He, Siddharth Barman, and Jeffrey F. Naughton. 2013. On Load Shedding in Complex Event Processing. arXiv:1312.4283 [cs.DB]
[30] Martin Hirzel, Robert Soulé, Scott Schneider, Buğra Gedik, and Robert Grimm. 2014. A Catalog of Stream Processing Optimizations. *ACM Comput. Surv.* 46, 4, Article 46 (March 2014), 34 pages. https://doi.org/10.1145/2528412
[31] Sepp Hochreiter and Jürgen Schmidhuber. 1997. Long Short-Term Memory. 9, 8 (Nov. 1997), 1735–1780. https://doi.org/10.1162/neco.1997.9.8.1735
[32] Steven C. H. Hoi, Doyen Sahoo, Jing Lu, and Peilin Zhao. 2021. Online Learning: A Comprehensive Survey. *Neurocomputing* 459 (2021), 249–289.
[33] MD. Zakir Hossain, Ferdous Sohel, Mohd Fairuz Shiratuddin, and Hamid Laga. 2019. A Comprehensive Survey of Deep Learning for Image Captioning. *ACM Comput. Surv.* 51, 6, Article 118 (Feb. 2019), 36 pages. https://doi.org/10.1145/3295748
[34] Zhiheng Huang, Wei Xu, and Kai Yu. 2015. Bidirectional LSTM-CRF Models for Sequence Tagging. arXiv:1508.01991 [cs.CL]
[35] Yiding Jiang, Behnam Neyshabur, Hossein Mobahi, Dilip Krishnan, and Samy Bengio. 2020. Fantastic Generalization Measures and Where to Find Them. In *8th International Conference on Learning Representations, ICLR 2020, Addis Ababa, Ethiopia, April 26-30, 2020*. OpenReview.net. https://openreview.net/forum?id=SJgIPJBFvH
[36] Ilya Kolchinsky. 2020. OpenCEP. https://github.com/ilya-kolchinsky/OpenCEP.
[37] Ilya Kolchinsky and Assaf Schuster. 2018. Efficient Adaptive Detection of Complex Event Patterns. *Proc. VLDB Endow.* 11, 11 (July 2018), 1346–1359. https://doi.org/10.14778/3236187.3236190
[38] I. Kolchinsky and A. Schuster. 2018. Join Query Optimization Techniques for Complex Event Processing Applications. *Proc. VLDB Endow.* 11 (2018), 1332–1345.
[39] Ilya Kolchinsky and Assaf Schuster. 2018. Join Query Optimization Techniques for Complex Event Processing Applications. 11, 11 (2018). https://doi.org/10.14778/3236187.3236189
[40] Ilya Kolchinsky and Assaf Schuster. 2019. Real-Time Multi-Pattern Detection over Event Streams. In *Proceedings of the 2019 International Conference on Management of Data* (Amsterdam, Netherlands) *(SIGMOD '19)*. Association for Computing Machinery, New York, NY, USA, 589–606. https://doi.org/10.1145/3299869.3319869
[41] Ilya Kolchinsky, Izchak Sharfman, and Assaf Schuster. 2015. Lazy Evaluation Methods for Detecting Complex Events. In *Proceedings of the 9th ACM International Conference on Distributed Event-Based Systems* (Oslo, Norway)

(DEBS '15). Association for Computing Machinery, New York, NY, USA, 34–45. https://doi.org/10.1145/2675743.2771832

[42] Eitan Kosman, Ilya Kolchinsky, and Assaf Schuster. April 2022. Mining Logical Arithmetic Expressions From Proper Representations. SIAM International Conference on Data Mining (SDM), Alexandria, Virginia, USA.

[43] John D. Lafferty, Andrew McCallum, and Fernando C. N. Pereira. 2001. Conditional Random Fields: Probabilistic Models for Segmenting and Labeling Sequence Data. In *Proceedings of the Eighteenth International Conference on Machine Learning (ICML '01)*. Morgan Kaufmann Publishers Inc., San Francisco, CA, USA, 282–289.

[44] Guillaume Lample, Miguel Ballesteros, Sandeep Subramanian, Kazuya Kawakami, and Chris Dyer. 2016. Neural Architectures for Named Entity Recognition. In *Proceedings of the 2016 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies*. Association for Computational Linguistics, San Diego, California, 260–270. https://doi.org/10.18653/v1/N16-1030

[45] Colin Lea, Rene Vidal, Austin Reiter, and Gregory D. Hager. 2016. Temporal Convolutional Networks: A Unified Approach to Action Segmentation. *arXiv* (Aug 2016). arXiv:1608.08242 https://arxiv.org/abs/1608.08242v1

[46] Jin Li, Kristin Tufte, Vladislav Shkapenyuk, Vassilis Papadimos, Theodore Johnson, and David Maier. 2008. Out-of-Order Processing: A New Architecture for High-Performance Stream Systems. *Proc. VLDB Endow.* 1, 1 (Aug. 2008), 274–288. https://doi.org/10.14778/1453856.1453890

[47] Qian Li, Hao Peng, Jianxin Li, Congying Xia, Renyu Yang, Lichao Sun, Philip Yu, and Lifang He. 2020. A Text Classification Survey: From Shallow to Deep Learning. *ResearchGate* (Aug 2020). https://www.researchgate.net/publication/343414448_A_Text_Classification_Survey_From_Shallow_to_Deep_Learning

[48] Zheng Li and Tingjian Ge. 2016. History is a Mirror to the Future: Best-Effort Approximate Complex Event Matching with Insufficient Resources. *Proc. VLDB Endow.* 10, 4 (Nov. 2016), 397–408. https://doi.org/10.14778/3025111.3025121

[49] Mo Liu, Ming Li, Denis Golovnya, Elke A. Rundensteiner, and Kajal Claypool. 2009. Sequence Pattern Query Processing over Out-of-Order Event Streams. In *2009 IEEE 25th International Conference on Data Engineering*. 784–795. https://doi.org/10.1109/ICDE.2009.95

[50] Zengjian Liu, Ming Yang, Xiaolong Wang, Qingcai Chen, Buzhou Tang, Zhe Wang, and Hua Xu. 2017. Entity recognition from clinical texts via recurrent neural network. *BMC Med. Inf. Decis. Making* 17, S2 (Jul 2017). https://doi.org/10.1186/s12911-017-0468-7

[51] Xuezhe Ma and Eduard Hovy. 2016. End-to-end Sequence Labeling via Bi-directional LSTM-CNNs-CRF. In *Proceedings of the 54th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*. Association for Computational Linguistics, Berlin, Germany, 1064–1074. https://doi.org/10.18653/v1/P16-1101

[52] Alessandro Margara, Gianpaolo Cugola, and Giordano Tamburrelli. 2013. *Towards automated rule learning for complex event processing*. Technical Report.

[53] Nijat Mehdiyev, Julian Krumeich, David Enke, Dirk Werth, and Peter Loos. 2015. Determination of Rule Patterns in Complex Event Processing Using Machine Learning Techniques. *Procedia Comput. Sci.* 61 (Jan 2015), 395–401. https://doi.org/10.1016/j.procs.2015.09.168

[54] Yuan Mei and Samuel Madden. 2009. ZStream: A Cost-Based Query Processor for Adaptively Detecting Composite Events. In *Proceedings of the 2009 ACM SIGMOD International Conference on Management of Data* (Providence, Rhode Island, USA) *(SIGMOD '09)*. Association for Computing Machinery, New York, NY, USA, 193–206. https://doi.org/10.1145/1559845.1559867

[55] Shervin Minaee, Nal Kalchbrenner, Erik Cambria, Narjes Nikzad, Meysam Chenaghlu, and Jianfeng Gao. 2021. Deep Learning Based Text Classification: A Comprehensive Review. arXiv:2004.03705 [cs.CL]

[56] openspecs sql. 2020. [MS-CEPM]: Glossary. https://docs.microsoft.com/en-us/openspecs/sql_server_protocols/ms-cepm/9b21b33f-af9f-41bb-9bf6-2b29e4579edc

[57] A. Ozbayoglu, M. U. Gudelek, and Omer Berat Sezer. 2020. Deep Learning for Financial Applications : A Survey. *Appl. Soft Comput.* 93 (2020), 106384.

[58] Rrubaa Panchendrarajan and Aravindh Amaresan. 2019. Bidirectional LSTM-CRF for Named Entity Recognition. *ResearchGate* (May 2019). https://www.researchgate.net/publication/333384813_Bidirectional_LSTM-CRF_for_Named_Entity_Recognition

[59] Razvan Pascanu, Tomas Mikolov, and Yoshua Bengio. 2013. On the Difficulty of Training Recurrent Neural Networks. In *Proceedings of the 30th International Conference on International Conference on Machine Learning - Volume 28* (Atlanta, GA, USA) *(ICML'13)*. JMLR.org, III–1310–III–1318.

[60] Antreas Pogiatzis and Georgios Samakovitis. 2020. Using BiLSTM Networks for Context-Aware Deep Sensitivity Labelling on Conversational Data. *Appl. Sci.* 10, 24 (Dec 2020), 8924. https://doi.org/10.3390/app10248924

[61] Olga Poppe, Chuan Lei, Salah Ahmed, and Elke A. Rundensteiner. 2017. Complete Event Trend Detection in High-Rate Event Streams. In *Proceedings of the 2017 ACM International Conference on Management of Data* (Chicago, Illinois, USA) *(SIGMOD '17)*. Association for Computing Machinery, New York, NY, USA, 109–124. https://doi.org/10.1145/3035918.3035947

[62] A. Qayyum, J. Qadir, M. Bilal, and A. Al-Fuqaha. 2021. Secure and Robust Machine Learning for Healthcare: A Survey. *IEEE Reviews in Biomedical Engineering* 14 (2021), 156–180. https://doi.org/10.1109/RBME.2020.3013489

[63] Yanmin Qian, Tian Tan, and Dong Yu. 2016. Neural Network Based Multi-Factor Aware Joint Training for Robust Speech Recognition. *IEEE/ACM Transactions on Audio, Speech, and Language Processing* 24, 12 (2016), 2231–2240. https://doi.org/10.1109/TASLP.2016.2598308

[64] Ella Rabinovich, Opher Etzion, and Avigdor Gal. 2011. Pattern Rewriting Framework for Event Processing Optimization. In *Proceedings of the 5th ACM International Conference on Distributed Event-Based System* (New York, New York, USA) *(DEBS '11)*. Association for Computing Machinery, New York, NY, USA, 101–112. https://doi.org/10.1145/2002259.2002277

[65] Tilmann Rabl, Jonas Traub, Asterios Katsifodimos, and Volker Markl. 2016. Apache Flink in current research. *it - Information Technology* 58 (01 2016). https://doi.org/10.1515/itit-2016-0005

[66] G. Ramena, D. Nagaraju, S. Moharana, D. Prasanna Mohanty, and N. Purre. 2020. An Efficient Architecture for Predicting the Case of Characters using Sequence Models. In *2020 IEEE 14th International Conference on Semantic Computing (ICSC)*. 174–177. https://doi.org/10.1109/ICSC.2020.00035

[67] Nils Reimers and Iryna Gurevych. 2017. Optimal Hyperparameters for Deep LSTM-Networks for Sequence Labeling Tasks. *ArXiv* abs/1707.06799 (2017).

[68] Doyen Sahoo, Quang Pham, Jing Lu, and Steven C. H. Hoi. 2018. Online Deep Learning: Learning Deep Neural Networks on the Fly. In *IJCAI*.

[69] Hasim Sak, Andrew W. Senior, and Françoise Beaufays. 2014. Long Short-Term Memory Based Recurrent Neural Network Architectures for Large Vocabulary Speech Recognition. *CoRR* abs/1402.1128 (2014). http://arxiv.org/abs/1402.1128

[70] Guy Shapira, Ilya Kolchinsky, and Assaf Schuster. 2022. Semi-supervised Frequent Pattern Mining for CEP. (2022). Manuscript paper.

[71] Gal Sidi, Ilya Kolchinsky, and Assaf Schuster. 2022. DELETE: Using deep learning to minimize latency in CEP systems. (2022). Manuscript paper.

[72] Mehmet Ulvi Simsek, Feyza Yildirim Okay, and Suat Ozdemir. 2021. A deep learning-based CEP rule extraction framework for IoT data. *J. Supercomput.* (Jan 2021), 1–30. https://doi.org/10.1007/s11227-020-03603-5

[73] Hadar Sivan, Mickey(Moshe) Gabel, and Assaf Schuster. 2020. Incremental Sensitivity Analysis for Kernelized Models. ECML-PKDD.

[74] Hadar Sivan, Mickey(Moshe) Gabel, and Assaf Schuster. 2022. AutoMon: Automatic Distributed Monitoring for Arbitrary Multivariate Functions. SIGMOD, Philadelphia, PA, USA.

[75] Ahmad Slo, Sukanya Bhowmik, Albert Flaig, and K. Rothermel. 2019. pSPICE: Partial Match Shedding for Complex Event Processing. *2019 IEEE International Conference on Big Data (Big Data)* (2019), 372–382.

[76] A. Slo, S. Bhowmik, and K. Rothermel. 2020. State-Aware Load Shedding from Input Event Streams in Complex Event Processing. *IEEE Transactions on Big Data* (2020), 1–1. https://doi.org/10.1109/TBDATA.2020.3047438

[77] Utkarsh Srivastava and Jennifer Widom. 2004. Flexible Time Management in Data Stream Systems. In *Proceedings of the Twenty-Third ACM SIGMOD-SIGACT-SIGART Symposium on Principles of Database Systems* (Paris, France) *(PODS '04)*. Association for Computing Machinery, New York, NY, USA, 263–274. https://doi.org/10.1145/1055558.1055596

[78] Ruoyu Sun. 2019. Optimization for deep learning: theory and algorithms. *ArXiv* abs/1912.08957 (2019).

[79] Charles Sutton and Andrew McCallum. 2012. An Introduction to Conditional Random Fields. *Found. Trends Mach. Learn.* 4, 4 (April 2012), 267–373. https://doi.org/10.1561/2200000013

[80] Chuanqi Tan, Fuchun Sun, Tao Kong, Wenchang Zhang, Chao Yang, and Chunfang Liu. 2018. A Survey on Deep Transfer Learning. *ArXiv* abs/1808.01974 (2018).

[81] K. Tawsif, J. Hossen, J. E. Raja, M. Z. H. Jesmeen, and E. M. H. Arif. 2018. A Review on Complex Event Processing Systems for Big Data. In *2018 Fourth International Conference on Information Retrieval and Knowledge Management (CAMP)*. 1–6. https://doi.org/10.1109/INFRKM.2018.8464787

[82] Elena Tutubalina and Sergey Nikolenko. 2017. Combination of Deep Recurrent Neural Networks and Conditional Random Fields for Extracting Adverse Drug Reactions from User Reviews. *J. Healthcare Eng.* 2017 (Sep 2017). https://doi.org/10.1155/2017/9451342

[83] Guy Uziel. 2019. Deep Online Learning with Stochastic Constraints. *CoRR* abs/1905.10817 (2019). arXiv:1905.10817 http://arxiv.org/abs/1905.10817

[84] Cheng Wang, Haojin Yang, and Christoph Meinel. 2018. Image Captioning with Deep Bidirectional LSTMs and Multi-Task Learning. *ACM Trans. Multimedia Comput. Commun. Appl.* 14, 2s, Article 40 (April 2018), 20 pages. https://doi.org/10.1145/3115432

[85] Yongheng Wang, Hui Gao, and Guidan Chen. 2018. Predictive complex event processing based on evolving Bayesian networks. *Pattern Recognit. Lett.* 105 (Apr 2018), 207–216. https://doi.org/10.1016/j.patrec.2017.05.008

[86] K. R. Weiss, T. Khoshgoftaar, and Dingding Wang. 2016. A survey of transfer learning. *Journal of Big Data* 3 (2016), 1–40.

[87] Eugene Wu, Yanlei Diao, and Shariq Rizvi. 2006. High-Performance Complex Event Processing over Streams. In *Proceedings of the 2006 ACM SIGMOD International Conference on Management of Data* (Chicago, IL, USA) *(SIGMOD '06)*. Association for Computing Machinery, New York, NY, USA, 407–418. https://doi.org/10.1145/1142473.1142520

[88] Yinjun Wu, E. Dobriban, and Susan B. Davidson. 2020. DeltaGrad: Rapid retraining of machine learning models. In *ICML*.

[89] T. Xing, M. Roig Vilamala, L. Garcia, F. Cerutti, L. Kaplan, A. Preece, and M. Srivastava. 2019. DeepCEP: Deep Complex Event Processing Using Distributed Multimodal Information. In *2019 IEEE International Conference on Smart Computing (SMARTCOMP)*. 87–92. https://doi.org/10.1109/SMARTCOMP.2019.00034

[90] Maor Yankovitch, Ilya Kolchinsky, and Assaf Schuster. 2022. HYPERSONIC: A Hybrid Parallelization Approach for Scalable Complex Event Processing. SIGMOD 2022, Philadelphia, PA, USA.

[91] Kyongsik Yun, Alexander Huyen, and Thomas Lu. 2018. Deep Neural Networks for Pattern Recognition. arXiv:1809.09645 [cs.CV]

[92] Zarita Zainuddin and Ong. 2008. Function approximation using artificial neural networks. *WSEAS Transactions on Mathematics* 7, 6 (Jun 2008). https://www.researchgate.net/publication/228840414_Function_approximation_using_artificial_neural_networks

[93] Haopeng Zhang, Yanlei Diao, and Neil Immerman. 2014. On Complexity and Optimization of Expensive Queries in Complex Event Processing. In *Proceedings of the 2014 ACM SIGMOD International Conference on Management of Data* (Snowbird, Utah, USA) *(SIGMOD '14)*. Association for Computing Machinery, New York, NY, USA, 217–228. https://doi.org/10.1145/2588555.2593671

[94] Haopeng Zhang, Yanlei Diao, and Neil Immerman. 2014. On Complexity and Optimization of Expensive Queries in Complex Event Processing. In *Proceedings of the 2014 ACM SIGMOD International Conference on Management of Data* (Snowbird, Utah, USA) *(SIGMOD '14)*. Association for Computing Machinery, New York, NY, USA, 217–228. https://doi.org/10.1145/2588555.2593671

[95] B. Zhao. 2018. Complex Event Processing under Constrained Resources by State-Based Load Shedding. *2018 IEEE 34th International Conference on Data Engineering (ICDE)* (2018), 1699–1703.

[96] Qunzhi Zhou, Yogesh Simmhan, and Viktor Prasanna. 2012. Incorporating Semantic Knowledge into Dynamic Data Processing for Smart Power Grids. In *Proceedings of the 11th International Conference on The Semantic Web - Volume Part II* (Boston, MA) *(ISWC'12)*. Springer-Verlag, Berlin, Heidelberg, 257–273. https://doi.org/10.1007/978-3-642-35173-0_17