January 2021

SYSTEM FOR EXPERT-ASSISTED CAUSAL INFERENCE FOR RANKING EVENTS OF INTEREST IN NETWORKS

Dmitry Goloubew

Nassim Benoussaid

Carlos M. Pignataro

Vladimir Yashin

SYSTEM FOR EXPERT-ASSISTED CAUSAL INFERENCE FOR RANKING EVENTS OF INTEREST IN NETWORKS

AUTHORS:
Dmitry Goloubew
Nassim Benoussaid
Carlos M. Pignataro
Vladimir Yashin

ABSTRACT

Networks have increased in size and complexity such that the number of events occurring each day has grown drastically. Techniques of this proposal provide for the ability to infer candidates for causal relationships—in some cases, with confidence. In particular, a novel machine learning (ML) based system is described that provides for the ability to narrow-down candidate temporal patterns that may potentially explain an event of interest (e.g., a network outage). The system is trainable with a human in the loop and is highly effective even with minimal amount of prior training.

DETAILED DESCRIPTION

Modern big data monitoring systems often produce alerts, alarms, and notifications that far exceed the attention and time budget of network operators to monitor or analyze. This inability to keep up with system events is a major impediment to adoption of automated monitoring and assurance systems.

This proposal provides a system that provides for the ability to consume the ever-growing amount of alerts generated on a network device and infer candidates for causal relationships. The system assumes to receive, as an input, a list of identified alert events / anomalies, from one or more devices over a period of time. After consumption of the identified alert events/anomalies, the system can returns a ranked list of potential causal relationships ranked by likelihood. The system then allows a Subject Matter Expert (SME) to review the ranked list of causal relationships in order to extract validated rules that are ready to be applied by the system ready.

A novel feature of these techniques lies in the combination of a machine learning anomaly detection system combined with an unsupervised causality inference system to

apply subject matter expertise. None of those components may be considered novel on their own.

Broadly during operation, the system may operate as follows:

1) An anomaly detection system identifies each outlier in a specific system at a certain time. Any anomaly detection system may be implemented (e.g., syslog alerts, alerts generated by manual scripts, unsupervised techniques leveraging machine learning (ML), etc.). In one instance, for example, such an anomaly detection system may include a Command Line Interface (CLI) output anomaly detection engine coupled with another tool to uniquely identify anomalies. For a CLI-based detection engine, each time CLI outputs are collected, new data points are created with the identified anomalies present in the output along with an associated timestamp.
2) All the data points are represented in a matrix in which each line of the matrix corresponds to a specific anomaly identity (ID).

Introduction and Details

Understanding causal relationships between discreet events would prove invaluable for network troubleshooting and remediation. A host of techniques can be employed for purpose of temporal pattern mining, however, bringing a number of identified candidate patterns has proven difficult and staggers adoption of said techniques for network troubleshooting and root cause analysis.

This proposal provides a novel approach to ranking candidate temporal patterns by embedding an operator in the loop. Having an operator label candidate patterns may provide multiple benefits, such as:

- Providing for the ability to build and maintain closed-loop system that can iteratively improve an embedded Learn-to-rank algorithm, which is a critical component atop of pure temporal pattern mining; and
- Allowing the ability to distill the most impactful identified patterns into a database of signatures that are prime for automated remediation at faster-than-human speed

A computer network is an ample source of events, both continuous and discrete. One component of fault analysis and troubleshooting involves finding causal links between

those, which is a laborious (and expensive) process, however, only a full understanding of said events and their causal relationship allows for the derivation of an action plan for remediation as well as set of measures that can be used to prevent an outage for occurring again.

Although full automation of a troubleshooting process has been eluding researchers for decades, this is still a process that can benefit from augmentation of human efforts using contemporary ML techniques.

For simplicity, consider that time may be treated as a sequence of discrete "time slices" in which every event can either be present or not in each slice. Events of interest that last for a period of time can therefore be presented as a contiguous sequence of discrete impressions, as shown below in Figure 1.

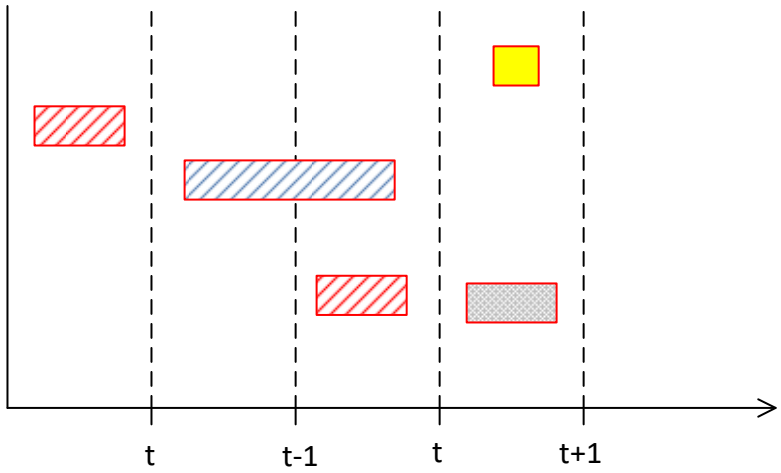

Figure 1: Example Event/Time Slice Plot

Different event types, such as "rate of incoming Border Gateway Protocol (BGP) update with anomaly score >1," "syslog message related to free memory depletion," "process crash," and/or the like may be represented for the plot as shown in Figure 1.

However, the number of events that may occur in a system may be astronomical and even number of event types is barely manageable. Yet, only a few events present may be of interest to a network operator. Typically, only events that show direct impact to service or business metrics are considered and are investigated, such as, network device reload, process crash, traffic or service disruption, etc.

Part of the troubleshooting process is analyzing the past (e.g., syslog journals, metrics monitoring plots, telemetry abnormalities) and finding precursor events that expose

underlying process(es) leading to failure hours or even days prior to such a failure. However, observing consistent sequence of events and/or a robust temporal pattern does not, by itself prove, causation between, two or more, constituent events. Yet, domain expertise may be useful when analyzing such data.

In some instances, however, even supplying meaningful chains of correlated events is difficult. There are many approaches to temporal pattern mining that use statistical methods to identify the most robust patterns that remain stable across network segments, time frames, configurations, etc. The simplest approach is to enumerate all possible patterns assuming "any event can lead to any other," create a co-occurrence matrix, and prune links that are not robust enough.

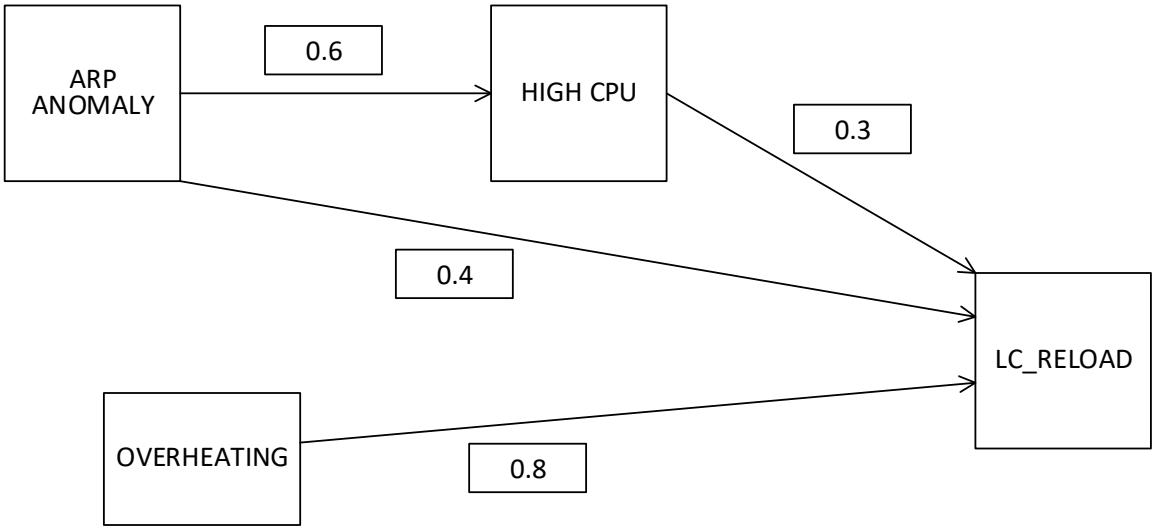

Figure 2: Example Event Patterns

Consider an example, as shown in Figure 2 above, which involves a demonstrated frequency approach to temporal pattern mining. For the example of Figure 2, a list of "candidate patterns" are shown where the Overheating -> LC_Reload (Line Card reload) pattern has the highest score, which can be explained by frequency and a low penalty for a "short" pattern.

There are more sophisticated ways to develop a list of candidate patterns (e.g., using recurrent neural networks such as Long Short-Term Memory (LSTM)), but they all suffer from the curse of dimensionality in that there will always be tens of thousands candidate patterns for network events, even when considering relatively simple networks.

Techniques herein seek to extend the above approach and provide an end-to-end system for temporal pattern mining that can be useful for workflow augmentation during troubleshooting processes, as illustrated in Figure 3, below.

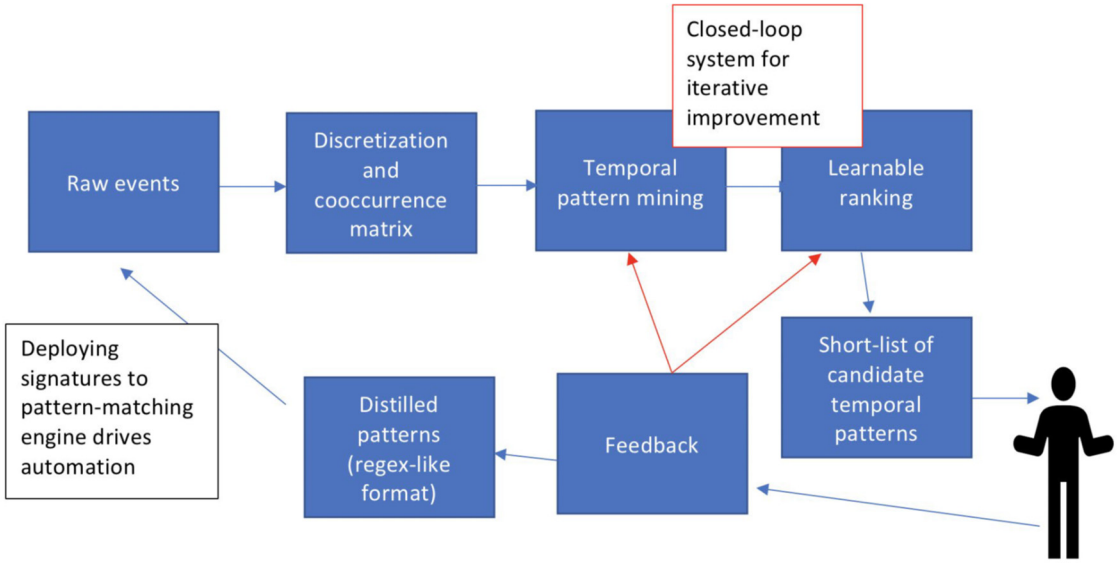

Figure 3: Example Workflow

For the example workflow as illustrated in Figure 3, any existing algorithms for temporal pattern mining may be utilized, but the choice of algorithms can be limited to those that employ optimization theory to optimize a loss function. Thus, the quality of the pattern-mining algorithm can be iteratively improved using signals derived in later steps.

The system may include a learn-to-rank component that ranks patterns derived from the pattern-mining algorithm according to a fitness function. One key is that the ranking component is "trainable" given enough feedback from a user.

Once patterns are ranked, an operator can investigate important events that have caused disruption to normal operation of the network, and the operator can be provided with recommendations for highly ranked candidate patterns.

The operator can reject patterns as irrelevant or not useful for investigation but can also promote other patterns as promising and/or insightful. This is a user interface (UI) - driven workflow in which feedback is captured in the form of a "training dataset" for ML algorithm training.

Thus, a closed-loop system is provided in which feedback from the operator is fed back into the system in order to improve pattern mining and ranking processes. Since both mining and ranking systems are fully differentiable, their performance can be improved by refitting their respective loss functions on derived dataset.

Simultaneously, candidate patterns that were flagged as useful can be "distilled" into a format suitable for a pattern-matching system in which actionable temporal signatures can drive automation for problem resolution. This provides for the ability to maintain a curated list of signatures for detection of important temporal patterns. Temporal pattern matching against such a list may not involve outrageous computational resources that are often associated with, for example, deep learning, while deep learning models can in fact be employed during the temporal mining step.

For such the workflow, consider an example in which different events may be detected at different times, such as:

T=t-2

- High CPU (anomaly)
- High rate of ARP traffic (anomaly)
- High rate of BGP updated (anomaly)
- T=t-1
- High CPU (anomaly)
- Low memory (syslog)
- High rate of BGP updated (anomaly)

T=t

- BGP process crashes (syslog)
- High rate of Address Resolution Protocol (ARP) traffic

Using such detected events, a time-adjusted co-occurrence graph/matrix can be generated illustrating the following event patterns:

Highcpu-lowmem-bgpcrash

Highcpu-lowmem-high arp

Highcpu-high arp

Higharp-highcpu

Highbgp-low mem-high arp

… and other patterns of length up to N.

Consider, for example, that after a user identifies that an event of interest is "BGP process crash" patterns can be pruned that do not lead to the selected outcome. After applying a temporal pattern-mining algorithm, some of the patterns can further be removed (e.g., highcpu-higharp can removed since it conflicts with higharp-highcpu and violates laws of causality). A ranking algorithm can then re-rank temporal patterns and the user can be presented with a reduced number of patterns. Say, for example, that the user is presented with 5 patterns and user feedback leads to following patterns being reinforced: highbgp -> low memory -> BGP process crash.

Utilizing the user feedback, a JavaScript Object Notation (JSON) file can be generated that provides a "signature" for the identified chain of events. The JSON can be consumed by the Network Management System (NMS) such that the signature becomes part of an Intellectual Capital (IC)/Signature database. Further, gradient updates can be utilized in the ranking and mining components so that patterns and rankings may be improved in future iterations of the system.

Additional details are now provided that further describe features associated with event patterns and signatures. For example, conversion to signatures in the system may involve obtaining anomalies from system dealing with text-based data and converting them into regular expressions—for the reason that underlying data is still expected to be text, just the IC engine changes to more lightweight form (applying Regular Expressions in the average case needs less computation than applying a ML model). In the general case, it can be envisioned that a causal link can be established between 2 events coming from different systems.

For example, a causal link can be determined between a data plane anomaly based on processing of model driven telemetry data (e.g., due to excessive Cyclic Redundancy Check (CRC) errors on a link inside a switching fabric of a multi-chassis router system) and a system analyzing Syslog data (e.g., a BGP neighbor timed out towards the system with CRC errors). Creating causal connections regarding inferences of different systems

would provide for the ability to elevate the abstraction level of IC-made alerts and help to enable the gradual transition towards multi-device and network-level IC, without having to rewrite exiting IC.

In light of the above, when it comes to producing signatures, it is critical to capture the relationship between events (i.e., what is causal what is caused). The signature format should be suitable for the engine that applies eventual signatures. In some instances, this may be a regular expression, however for a general case, an IC engine should be benefit from causal knowledge. To enable this, many reasoning engines accept knowledge in form of <subject>, <predicate>, <object> triplets. For some systems, knowledge may also be accepted in a form of <cause_id>, <effect_id>. In some implementations, abductive reasoning may be utilized to determine event relationships, as such systems are able to reason in both directions (from cause to effect, and from effect to cause).

Consider further, an example in which a potentially anomalous pattern has been identified, as illustrated in Figure 4. There may be challenges with storing the pattern in an intermediate format that can be used for automatic detection of a same or similar event following the signature.

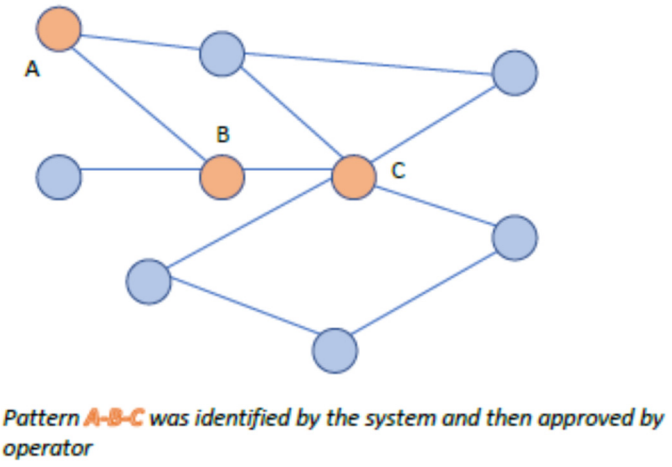

Figure 4: Example Anomalous Pattern

Once the pattern as illustrated in Figure 4 is approved, it can be stored in the IC database. Although techniques herein do not impose strict limits on *temporal pattern mining* algorithms to be employed for *candidate pattern selection*, it may be useful to transform raw patterns into a form that can be consumed by many algorithms given the

sheer complexity of modern algorithms. As illustrated in Figure 5, below, a single pattern may lead to multiple, domain-specific intermediate representations for different algorithms.

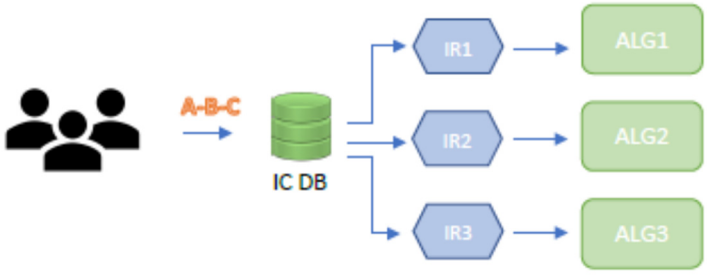

Figure 5: A Single Pattern May Give Rise to Multiple, Domain-Specific Intermediate Representations

Various intermediate representations (IRs) for some common algorithms may include:

1. A simple frequency matrix involving time-adjusted occurrence frequency, something any Relational Database Management System (RDBMS) can handle.
2. A Graph-CNN that can be retrained daily/weekly on entries in an IC DB. Such a deep neural network (NN) may be able to detect patterns identical or similar to A-B-C, once retrained.
3. Deep Learning models with external memory and Monte-Carlo Tree Search (MCTS) Reinforcement Learning systems can use an IC DB directly. However, export into suitable format is still required.
4. IR for Graph Space Embedding is essentially a long floating-point vector produced by Embedding NN (encoder). Algorithms for pattern matching can be reduced to a vector dot-product and can include encoding for A-B-C pattern to increase upfront speeds for matching since only the right-hand argument (*candidate pattern*) has to be encoded in search phase.
5. A Simple Regular Expression engine may use patterns from an IC DB directly, while a complex symbolic system, such as a Deep Fusion Reasoning Engine may need additional metadata, such as event type information.

It is to be understood that algorithms evolve over time and some IRs will inevitably become obsolete. Thus, storing an original pattern in the IC DB can help to ensure

operational continuity and can also allow for the plug-and-play nature of mining algorithms. Further, proper archival, versioning, inspection by, and expert/continuous testing of patterns stored in the IC DB remain possible.

Additionally, depending on the implementation and/or security implications, only some of the IRs may be distributed and used for detection if, for example, pattern mining is performed on a device outside of IC owner control and IC leakage is a concern. For example, an IR for a Graph Space Embedding network (floating point vector) can be distributed even in a partly adversarial environment where raw data is not safe to transmit and store. Expanding on privacy and encryption, a pure numerical IR of an IC signature may even allow matching through application of full homomorphic encryption.

As another example of event patterns, in some instances, it may be useful to represent events happening in a network as a graph, as shown in Figure 6, for cases in which two events share a unidirectional link from E1 to E2, if E1 could be the cause of E2.

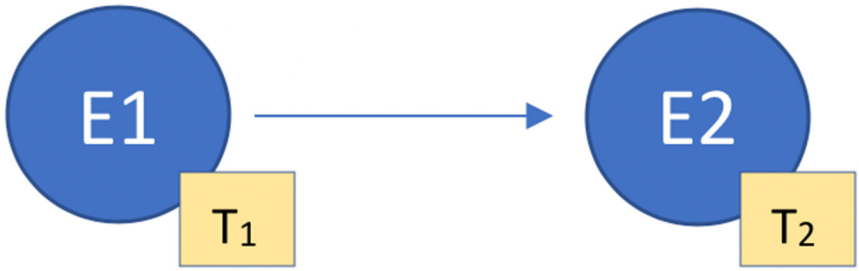

Figure 6: Example Event Graph

Apriori, making no assumptions about nature of the domain, all events can cause–and hence lead–to any other events with only time dependency holding the graph of events from becoming fully meshed. Indeed, event E2 that happened at time T_2 cannot be the cause of event E1 that happened at T_1 where $T_1 < T_2$. However, the "time" dependency involving prior events, although useful, still is not enough to determine relationships, since the number of connections would grow exponentially in number of events.

Thus, in order to prune the graph of events, techniques herein propose following algorithm:

- Operate on event sequences of length up to N that are part of common graph G. This ensures locality of decision making in time.
- Drop graph edges that statistically are unlikely to be causal judging from past history. Once the system is bootstrapped (by other means discussed herein) trivial operations can be performed to remove connections that are not causal with high certainty. For example, if "CPU load being high on router R1" is sometimes being observed prior to "traffic loss through router R2", but the causal link is disproved time and time again, this connection can safely be pruned from graph G the next time it is observed.
- Employ a Deep Fusion Reasoning Engine (DFRE) that uses a Non-axiomatic Reasoning System (NARS) to blend symbolic knowledge (in the simplest case, hand-written rules) and engage in decision-making similar to fuzzy logic. This means that operators may be allowed to annotate some of the links and the DFRE can provide filtering and prioritization capabilities to prune links from graph G.
- An operator identifies event of interest. For example, if an operator has identified event E as one in need of explanation we only need to consider causal chains that one way or another lead to node E. In a graph G, this can be done by exhaustive search of sequences of length up to M (A* search is a good candidate) or by randomly sampling start nodes from graph G until a time quota is exhausted.

Once graph G is made more sparse, potential causal chains can be ranked utilizing Learn-to-Rank (L2R) approaches, which typically treat ranking as a differentiable process that can be optimized by gradient descent methods (e.g., stochastic gradient descent) where a training signal is obtained from an interaction with an end user (e.g., network operator). There is rich corpus of literature on L2R techniques that are predominantly used to optimize ranking for searches in databases, the Internet, etc. in which metrics such as click-through ratio (CTR), page dwell-time, and others can be used as measure of engagement with content. One example of such system is a "Learning to Rank Question Answer Pairs with Holographic Dual LSTM Architecture."

In the approach of this proposal, potential causal chains are presented (as mined from graph G) via a user interface (UI), as shown below in Figure 7, through which operators can validate and provide feedback, which can be captured.

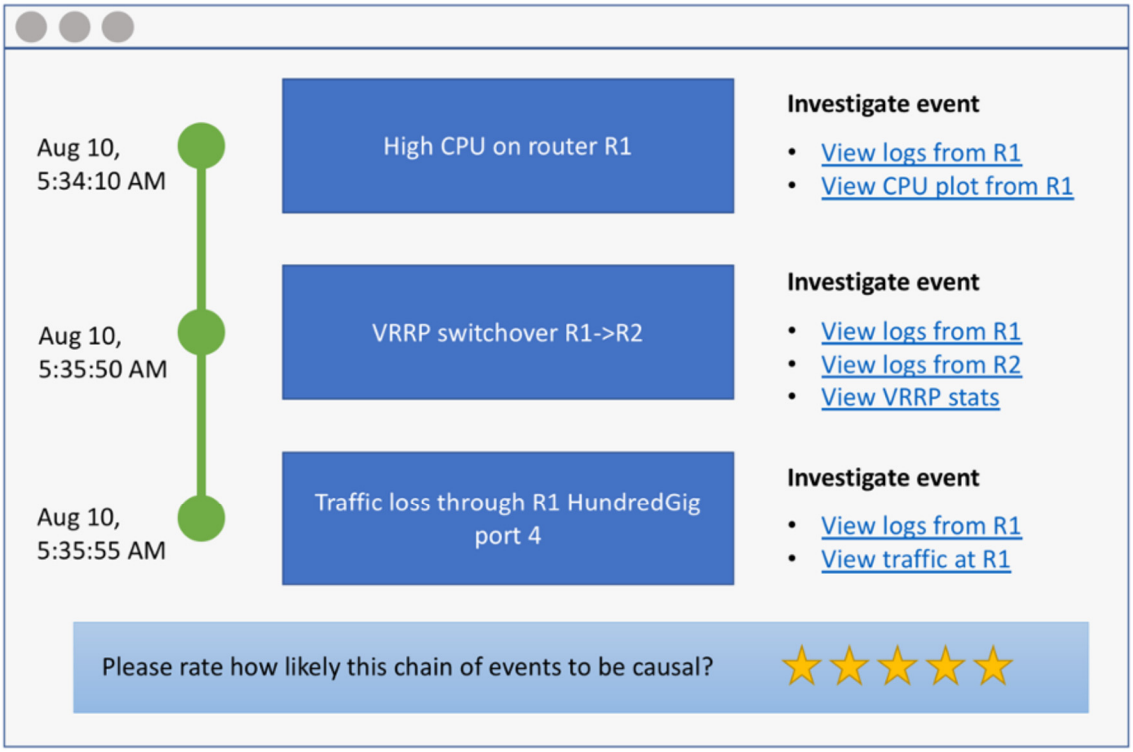

Figure 7: Example User Interface to Validate and Provide Feedback

In some instances, it may be useful to represent a chain of events of length M so that the chain of events could be used as input to a neural network. Since such a chain is essentially a subgraph of G, it is proposed to use one of many available graph-embedding methods such as those used in a Kernel Graph Convolutional Neural Network and Graph Space Embedding.

Individual events/nodes can be embedded respective to their type. For example, log messages, being mostly text, can be represented densely through a language model embedding (e.g., Generative Pre-trained Transformer 3 (GPT-3), Bidirectional Encoder Representations from Transformers (BERT), word2vec, etc.). Numerical and categorical values can be used directly. Once a critical amount of candidate causal chains (densely embedded) and respective feedbacks are amassed, the problem can be treated as a typical

statistical learning with deep neural networks. Additionally, annotated chains can be stored directly in structured rule-based format for consumption with a DFRE.

In comparison to other systems/solutions, a key mission of the capability described in this proposal is to facilitate the reduction of brittleness and bias in areas where training data is not sufficiently representative. Examples of this can include accidents for driverless cars, outages for networking systems, or, more generally, complex events/activities that are very hard or impossible to understand through many examples.

Consider a complex activity such as driving a car. Although seeing a few examples of a human driving may allow for the generalization that a human can drive a car, artificial systems may not be able to correctly identify such an activity being performed. Stated differently, observing a trained system correctly transcribe a picture of children playing ball on a lawn does not necessarily give any assurance that other pictures (having different or similar content) will be transcribed correctly.

In contrast, this proposal provides a system that can utilize upstream systems (e.g., monitoring/detection systems) as a source of events. These events are turned into causality-candidates that are shown to SMEs. The SMEs do not have to see all the underlying anomalies, which makes the use of SME time much more economical (and thus realistic). A SME can make the final determination as to whether a candidate represents a causal relationship.

Some solutions use expert annotated anomalies directly as training. This works well when SMEs generate a sufficient amount of annotations. However, a single annotation rarely provides for the ability to change the behavior sufficiently via retraining, which is another difference between this proposal and other solutions. This proposal utilizes model assertions, external reasoning, or rule systems to apply knowledge, thereby bypassing the retraining loop. This may be useful for cases in which experts may not generate sufficient feedback to change model behavior in a desired manner.

In summary, techniques herein provide a system that can receive, as inputs, raw events over time, which may be obtained from any anomaly detection and/or monitoring systems, potentially from different places in a network. Short-lists of candidate anomalous temporal patterns of a certain length can be provided to an expert reviewer through any form of UI and a feedback mechanism can be utilized to learn a better ranking of the

candidate anomalous temporal patterns each time an expert validates or discards patterns. Finally, the IC created using the expert assistance can be stored in various forms (Intermediate Representations) and can later be fed into a production anomaly detection system.