ABSTRACT
We live in a data abundance era. Availability of large volume of diverse multimedia data streams (ranging from video, to tweets, to activity, and to PM2.5) can now be used to solve many critical societal problems. Causal modeling across multimedia data streams is essential to reap the potential of this data. However, effective frameworks combining formal abstract approaches with practical computational algorithms for causal inference from such data are needed to utilize available data from diverse sensors. We propose a causal modeling framework that builds on data-driven techniques while emphasizing and including the appropriate human knowledge in causal inference. We show that this formal framework can help in designing a causal model with a systematic approach that facilitates framing sharper scientific questions, incorporating expert’s knowledge as causal assumptions, and evaluating the plausibility of these assumptions. We show the applicability of the framework in an important Asthma management application using meteorological and pollution data streams.

Categories and Subject Descriptors
H.4 [Information Systems Applications]: Miscellaneous; D.2.8 [Software Engineering]: Metrics—complexity measures, performance measures

General Terms
Causal Modeling, Multimedia, Multimodal Data Streams

Keywords
Causal Analysis, Causal Patterns, Asthma Risk Factor Recognition

1. INTRODUCTION
A common topic in multimedia content analysis is detecting events and understanding relationships among those events from video streams, audio streams, and text. More recently other data modalities such as wearable and environmental sensory streams have attracted attention in multimedia community. Detecting low-level events (e.g. tracked objects appearing in certain spatial relationships) and higher-level events (e.g. from surveillance videos and news) are well studied research topics in multimedia. Today with instantaneously accessible data from wireless sensors, biosensors, and mobile apps other fields such as medical research, biology, and physiology are watching up with it and are offering even more challenging problems in understanding relations between domain-level events from disparate streams. So we are facing a unique situation of abundance of multimedia data that can be used to solve challenging societal problems. Most of our existing techniques and concepts are rooted in the time when we had scarcity of data and limited computing and communication technology. But with the emergence of multimedia Big Data our challenge has shifted towards handling this abundance and harnessing the true potential of this ever-growing data. This data spreads over heterogeneous sources containing unstructured, semi-structured, or structured information. Before any analysis, unified signals need to be created from asynchronous, multivariate, and mixed-modality information. Then conceptual models are constructed to account for data in different application domains.

From signal processing perspective, sensory data streams are continuous numerical time-series. Signal models entail a mathematical representation of data so as to extract desirable information from signals. Signal models have been successfully used in various problems that include image processing, speech processing, signal filtering and prediction. These models generate encouraging results in low-level perceptual and signal processing tasks. However, their success is limited in higher cognitive areas where data streams are heterogeneous and symbolic modeling and reasoning are required.

Whenever one talks about models, prediction is presumed to be the end goal. However there are other benefits in creating models. Models explain a phenomenon, which is quite different from prediction. For example, in healthcare domain, a model can explain the relation between multiple asthma risk factors (i.e. air pollution, pollen, and weather fluctuation) and asthma attack and not necessarily predict when attack might happen. Also models raise new questions and in the processing of finding answers, interesting insight might be discovered. Most importantly models are used to
understand causes of a problem. So causal analysis is the artifact of models.

Inferring causal relationships is one of the central tasks of science, it is a topic that has been heavily debated in philosophy, statistics, and scientific disciplines. We all have a mental model of cause and effect that helps reasoning about daily life. Individuals prefer explanations for phenomena couched in terms of cause and effect [9]. Causes can be actions, states, or state transitions. So we use the general term event that include any potential cause or effect. In most applications there is not a simple or a linear causal relationship between events and the meaning of causal assertions depend on temporal constraints between a set of events.

Consider Fig. 1 as an example of causal relation between multiple data streams. The first data stream is measured from tweets, and depicts the burstiness of asthma topic in a specific geo-location (Tokyo city). Other streams are measured by physical sensors in the same location, and represents meteorological and pollution data. Once a burst in asthma related messages is detected, physical sensory data within a preceding TW time window is analyzed to find a set of complex causal patterns that might have resulted in the burst. From this figure two causal patterns can be inferred: 1) pollution increases suddenly followed by high wind while temperature increases slightly will cause an asthma outbreak. 2) A thunderstorm followed by temperature decreases steadily will cause an asthma outbreak. Of course, finding such patterns within one dataset doesn’t prove the validity of a causal model, but it lends credibility to it. If we repeatedly test this model on more data and in diverse related datasets (e.g. from other cities) and find good matches to the data, then the inferred causal model gain more credibility.

Building models involve many decisions such as determining model selection strategy, defining a model structure, defining criteria for model goodness, selection of data and transformation applied to it, just to name a few. Most of these decisions involve reliance on theoretical or empirical results, that is expert’s domain knowledge, and cannot be learned by a system itself solely from available input data. Moreover, causal inference is not simple. There is not an easy answer for problems where analyst wants to estimate a causal pathway between a sequences of causes while some minor causes might happen in parallel as well (will elaborate on this in section 2). So it is often necessary to incorporate human judgment into this modeling process. When we encounter large volumes of disparate information, data-driven discovery techniques are quite useful in refining human judgment. So when scientist doesn’t have any idea what hypothesis to generate or how to proceed with data analysis task, automatic data-driven techniques provide significant insight. By seeding a hypothesis based on this insight, analyst can incorporate her own domain knowledge and formulate a refined hypothesis quickly, systematically, and ‘grow’ a hypothesis iteratively to generate a comprehensive model. The former is called ‘unknown-unknown’ problem and the latter is called ‘known-unknown’ problem. The best practice is to create a balance between these two. We believe a comprehensive modeling technique should deep human in the loop while taking advantage from new data-driven modeling techniques.

1.1 Problems with Current Approaches

The avalanche of sensory data has led scientists to envision a future in which automated modeling techniques, or ‘data-driven discovery’ will eventually rival the traditional hypothesis-driven research that has dominated research areas for at least the past century. Most researchers are fascinated by deep learning to model high level abstractions in data. These methods are computationally expensive and there are many training parameters need to be considered. Also they still lack ways of representing causal relationships and have no obvious ways of performing logical inferences.

Situated with temporal data mining research domain, causal inference is reduced to finding significant serial or parallel temporal patterns in time-series data. It is assumed that if the frequency of a pattern is greater than a predefined threshold, the pattern is significant. In serial patterns, events occur in a specific order while there is no ordering between events in parallel patterns. Many algorithms are proposed to mine time points [16, 1] or time interval [19, 18] patterns. But handling incomplete and partial information has been less studied. Interestingly the majority of these algorithms deal with serial and parallel patterns separately. So cause-effect patterns cannot contain both sequential and parallel components simultaneously, as depicted in Fig. 3. Temporal data mining products (e.g. commercial software Theme 1) work as a black box. They don’t involve human in the analysis process and their input is limited to numerical time-series.

In general, bypassing the analyst and carrying out causal analysis as a black box via learning or statistical techniques have several problems:

- Hiding the inference from the analysts reduce their understanding of the model. Also lack of understanding might result in overconfidence, with potentially catastrophic consequences when learned models actually fail in real-world situations.

1http://www.noldus.com/human-behavior-research/products/theme
Available methods typically cannot give advise to the analyst as how to refine the model and what to do next. By using available products, it is easy to fall into two traps: either the system acts at a very high level, and it is difficult to deviate from the way the developers of the product thought was the proper way to approach the problem, or the system acts at a very low level, and the analyst has to know a great deal in order to use the system meaningfully. Changing even a small parameter in a learned causal model, either for specific circumstances or for including individual judgment, is difficult and needs the whole model to be re-learned. Even if the source code is available, the persons qualified to do the changes may not have the time to tap into the code successfully.

To tackle these problems, we propose an appropriately designed causal modeling platform with 3 principles in mind: 1) There should be a balance between data-driven and hypothesis-driven modeling techniques, when one complements the other. 2) Analyst needs to understand causal patterns descriptively. 3) For each causal hypothesis, analyst needs to perform a separate analysis which results in a causal model. At the end, multiple causal models describe the end goal of the analysis.

1.2 Brave New Idea

We present a causal modeling technique that facilitates deep computational understanding and model building in different domains using traditional as well as emergent new multimedia big data streams. It is a new tool for empirical research and offers a natural environment for the study of structural and temporal inter-relations between multiple heterogeneous data sources. The novelty of our approach are: 1) Abstracting data streams to conceptual events as a unified data representation that is understandable for experts and facilitates the manipulation of a model’s structure; 2) Design a high-level declarative language which allows for definition of complex causal patterns from the combination of a unique set of well-defined operators. The language is very concise, contains a small number of operators, and is still very expressive; 3) Provide data-driven processing techniques, using a combination of basic operators and visual analytics to generate an insight in unknown-unknown situations. 4) Design a framework and a comprehensive GUI that engage analysts in an interactive, online process, allows them to iteratively use their knowledge to generate and evaluate a causal hypotheses to build a model.

Fig. 2 demonstrates our evolving system’s UI. Analyst can import pre-processed data (i.e. event streams) and choose visual data-driven operators to explore data, generate a basic model and derive a preliminary insight. Then analyst can seed a hypothesis and grow it step by step using provided operators. A good hypothesis is not the one that is necessarily correct, but one that opens up a new path of investigation. In complex problem domains this path cannot be fully perceived in advance. So analyst must be provided with appropriate operators to carry out new analyses based on the original hypothesis.

The proposed causal modeling approach has the important effect of coupling both data-driven and hypothesis-driven modeling, keeping human in the loop while taking advantage of data-driven analytics when needed. In this sense, it is a new hybrid modeling approach. This is a general platform and can be applied to different problem domains. In the evaluation section we demonstrate the applicability of such platform in a healthcare application.

2. CAUSAL MODELING

The most popular data representation is N-dimensional numerical vectors. Hence, modeling is usually carried out in $R^N$ based on algebraic semantics. For non-numeric data such as survey entries or computer logs, the general practice is to transform them to $R^N$ and force a real vector observation into the problem. However, a data transformation to $R^N$ typically abandons original data semantics. Therefore, the ability to carry out computations beyond $R^N$ could be valuable for developing useful models while retaining original problem semantics. One one hand, data representation needs to have representational power to capture the necessary aspects of a complex model. On the other hand, it has to be understandable for experts to facilitate the manipulation of a model’s structure. Some complex representations might hinder such understanding and manipulation. As suggested in [5], in a pre-processing phase, different machine learning, discretization, statistical and mathematical, mapping, and natural language processing techniques can be used to convert raw heterogeneous measurements to events (and event’s attributes). These events are application dependent and abstract sensory data into a higher conceptual level. Then causal patterns can be formulated by applying a set of operators on these events.

Fig. 3 shows a sample complex causal pattern. Multiple ordered events with time lags in between generate a sequential causal pattern. Also some events are minor causes and when coupled with other events at the same time (e.g. occur in parallel) results in an effects, that is so called concurrent causal pattern. Bringing an example from asthma management problem, a causal pattern of asthma attack can be: $(e_1: \text{patient did not take medication}) \text{ within } 60 \text{ minutes followed by } e_2: \text{engaged in an intense exercise}) \text{ while } (e_3: \text{pollution is high}) \text{ or } (e_4: \text{temperature is below } 20^\circ C) \text{ within } 15 \text{ minutes followed by } (\text{asthma attack})$.

$$\omega_15 \in E5$$
We provide a high level causal pattern language that contains a set of operators to formulate such complex patterns. Bellow is the asthma attack causal pattern translated into events and operators. The semantics of these operators are explained in section 2.3.

2.1 Time Model

For the purpose of temporal reasoning Allen formalized temporal logic on intervals by specifying 13 interval relations [2] and showing their completeness. Any two intervals are related by exactly one of the relations. The operators are: before, meets, overlaps, starts, during, finishes, the corresponding inverses after, met by, overlapped by, started by, contains, finished by, and equals. These temporal relations have been used by the majority of research on mining time-interval data [12, 14, 6, 3, 6]. However, researchers identified problems using Allen’s relations such as ambiguousness in the pattern representation because the same pattern can describe different situations in the data, and lack of robustness to noise because small shifts of time points lead to different patterns [15, 19]. After Allen, Freksa revisited interval relationships at the semi-interval level [4]. He generalized the interval relations by using semi-intervals with the following 11 operators: older, younger, head to head, survives, survived by, tail to tail, precedes, succeeds, contemporary, born after death, and died before birth. Semi-intervals allow a flexible representation where partial or incomplete knowledge need to be handled since operations are on parts of an interval and not the whole.

In temporal data mining, very little work has focused on using semi-intervals. The most notable was conducted by Morchen et al. in [13] where they explored semi-intervals for unsupervised pattern mining and proved that semi-interval patterns are more flexible than patterns over full intervals. In this respect, authors proposed a symbolic interval representation that can include complete intervals or only the starting and ending time points expressing a mixture of intervals and semi-intervals. They argued that by relaxing the constraint that the complete interval must be observed the patterns are more flexible in matching similar situations in data. In our work, we use semi-interval to represent temporal facet of an event. We adopt a time point representation of time intervals [19] in which intervals are represented with their start and end time point.

Definition 1 (Time Domain). A time domain $\mathbb{T}$ is a discrete, linearly ordered, countably infinite set of time instants $t \in \mathbb{T}$. We assume that $\mathbb{T}$ is bounded in the past, but not necessarily in the future.

Definition 2 (Time Interval). A time interval is a triple $[\partial, t_s, t_e]$ where $\partial \in \Sigma$ is a unique symbol and $t_s, t_e \in \mathbb{T}$ and $t_s \leq t_e$. The finite set of all time intervals is noted $I = \{[t_s, t_e] | t_s \leq t_e \}$. If $[t_s, t_e] \cap [t_s', t_e'] = \emptyset$, intervals are overlapped.

Definition 3 (Semi Interval). A semi interval is a tuple $[\partial^+, \partial^-] t \partial \cap \Sigma \Sigma$ is a unique symbol and interval boundaries are represented with + and signs where $\partial^+$ and $\partial^-$ are corresponding to the start and end of the interval respectively. $[\partial^+, t], [\partial^-, t]$ represent a semi interval where start time is available and $[\partial^-, t]$ represent a semi interval where end time is available. Using the semi-interval definition, an interval can also be represented with its interval boundaries. Formally in the time interval $[\partial, t_s, t_e], \partial t_s = \partial^+$ and $\partial t_e = \partial^-$ and duration of the interval is $d(\partial) = \partial^+ - \partial^-$. Also an instantaneous time point can be considered as an interval with zero duration where $\partial^+ = \partial^-$. The most important criteria in any cause-effect pattern (as depicted in Fig. 3) is that an effect always occurs after a set of causes within a time lag. Two situations need to be considered: 1) The causes occur in a particular order with specific time lag in between. 2) The order is not important but there has to be a notion of concurrency.

We define two temporal relations to meet these requirements. Then we use semi-interval operators to represent these temporal relations. The relations as shown in Table 1 are:

1. Order: is the sequential occurrence of time points or time intervals. In Freksa’s formalism younger, succeeds, survives, and born after death relations fall in this category.
2. Concurrency: is a nonempty time period where two or more temporal events occur in no particular order. In Freksa’s formalism head to head, tail to tail, and contemporary relations fall in this category.

![Figure 3: A sample structure of multiple causes and an effect. Causes might happen in a sequential order or concurrent.](image)

Table 1: Semi-interval relations

<table>
<thead>
<tr>
<th>Temporal Relation</th>
<th>Temporal Operation</th>
<th>Symbol</th>
<th>Semi-interval Representation</th>
<th>Graphical Illustration</th>
</tr>
</thead>
<tbody>
<tr>
<td>Order</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Y is younger than X</td>
<td>yo</td>
<td>$X^+ &lt; Y^-$</td>
<td>$\mathbb{X\ldots X}$</td>
<td></td>
</tr>
<tr>
<td>Y survives X</td>
<td>s</td>
<td>$X^- &lt; Y^-$</td>
<td>$\mathbb{X\ldots X}$</td>
<td></td>
</tr>
<tr>
<td>Y succeeds X</td>
<td>d</td>
<td>$X^- &lt; Y^-$</td>
<td>$\mathbb{X\ldots X}$</td>
<td></td>
</tr>
<tr>
<td>Y died after birth</td>
<td>dhab</td>
<td>$X^+ &lt; Y^-$</td>
<td>$\mathbb{X\ldots X}$</td>
<td></td>
</tr>
<tr>
<td>Concurrency</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Y is head to head with X</td>
<td>hh</td>
<td>$X^+ = Y^+$</td>
<td>$\mathbb{X\ldots X}$</td>
<td></td>
</tr>
<tr>
<td>Y is tail to tail with X</td>
<td>tt</td>
<td>$X^- = Y^-$</td>
<td>$\mathbb{X\ldots X}$</td>
<td></td>
</tr>
<tr>
<td>Y is contemporary with X</td>
<td>ct</td>
<td>$X^+ \cap Y \neq \emptyset$</td>
<td>$\mathbb{X\ldots X}$</td>
<td></td>
</tr>
</tbody>
</table>

2.2 Event Model

An event is either an instantaneous occurrence or spans over time. The former is called point event and the latter is called interval event. Sometimes we have a semi-interval event where information about the event is not complete. For example we might know when event starts but there is no information available about when it ends or vice versa. Event model has a schema $S$ that describes a set of attributes a class of events must contain.
Definition 3 (Point Event). A point event \((pE)\) is an event that occurs at an instantaneous point in time. It is a tuple \(e = (v, [E, t])\) consists of the name of its type, denoted by an upper-case letter (e.g., \(E\)), the time of occurrence \(t \in T\), and a set of values \(v \in S\).

Definition 4 (Interval Event). An interval event \((iE)\) is an event that spans over time. It is a tuple \(e = (v, [E, t_s, t_e])\) consists of the name of its type, start and end time \(t_s, t_e \in T\), and a set of values \(v \in S\).

Definition 5 (Semi-interval Event). A semi-interval event \((sE)\) is a special case of interval event when one of the event boundaries is missing. It is a tuple \(e = (v, [E^{+/−}, t])\) consists of the name of its type, start time or end time \(t \in T\), and a set of values \(v \in S\).

These three categories of events with their graphical illustration are shown in Table 2. Each event has a start time value \(t_s\) or an end time value \(t_e\) or both. Events with complete time interval are represented as \((E, t_s, t_e)\), while events with semi-interval are represented as \((E^{+/−}, t)\) when \(t_s\) is available, and \((E^{−/+}, t_e)\) when \(t_e\) is available. Point events are represented as \((E, t)\).

The reason that we differentiate between semi-interval event and point event goes to different semantics of events in different application domains. Semi-interval event is an interval event when a partial knowledge or observation is available. For example in healthcare domain when a patient ‘experience symptoms’ after taking a medication, this process is an interval event with start and end time, but we might only know when the symptoms start. So experiencing symptoms is recorded as a semi-interval event with the start time associated with it. In the same context, ‘taking a pill’ event is a point event. Though we can imagine that the duration of this event is several seconds, but with respect to the basic time granularity of an application (e.g., 15 minutes or 1 hour), ‘taking a pill’ is considered as a point event.

Definition 6 (Event Stream). An event stream \(ES^{(i)} = \{e^{(i)}_1, e^{(i)}_2, ..., e^{(i)}_n\}\) is an ordered set of events where \(e_k \in \{pE, iE, sE\}(1 \leq k \leq n)\).

Definition 7 (Multi-event Stream). A multi-event stream \(ES = \{ES^{(1)}, ES^{(2)}, ..., ES^{(l)}\}\) is a finite set of event streams. Events from multiple event streams have a total order based on their start time so they can be overlapped. The alphabet of multi-event stream is \(\Sigma = \bigcup \{\Sigma^{(1)} \cup \Sigma^{(2)} \cup ... \cup \Sigma^{(l)}\}\).

Take an example from Fig. 4(a), \(ES^{(1)}\) is an event stream with 3 disparate events. \(E_1\) is an interval event represented with \(E_1\) for start and end points, \(E_2\) is a point event where \(E_2 = E_1\) and \(E_3\) is a semi-interval event represented with \(E_3\) for start point. The events that belong to the same event stream cannot have overlaps. However, as shown in Fig. 4(b), events from multiple event streams might be overlapping. This brings us to the next question: How to encode a multi-event stream with overlapping events so the ordering and temporal structures between events are preserved?

To encode a multi-event stream with overlapping events so the ordering and temporal structures between events are preserved?

2.3 Processing Operators

We formally define our language by defining an operation algebra for causal pattern formulation. These operators are aimed to be the basic operations, combination of which can be used for arbitrarily sophisticated pattern formulation and pattern querying on event streams.

To begin with, each event type \(E_i \in \{pE, iE, sE\}\) is a pattern expression. The semantic of base expression for point events \(pE\), represented as \(E_i(t)\), is at a given time point \(t\), \(E_i(t)\) is true if \(E_i\) occurs at time point \(t\). The semantic of base expression for interval events, represented as \(E_i(t_s, t_e)\), is at a given time interval \(T = [t_s, t_e]\), \(E_i(t_s, t_e)\) is true if \(E_i\) starts at \(t_s\) and ends at \(t_e\). The semantic of base expression for semi-interval events, represented as \(E^{+/−}_i = E^{+/−}_i(t)\), is at a given time \(t\), \(E^{+/−}_i(t)\) is true if \(E_i\) starts at \(t\) and \(E^{+/−}_i(t)\) is true if \(E_i\) ends at \(t\).

Arbitrary patterns can be defined by applying operators on individual event types. Each pattern consists of a number of participating events. Suppose pattern \(\rho\) has \(k\) participating events \(E_1, \ldots, E_k\), size of the pattern denotes \(|\rho| = k\), start and end timestamps of the pattern denotes \(\rho.t_s\) and \(\rho.t_e\) respectively and defined as follows:

\[\rho^+ = \rho.t_s = \min\{E_1.t_s, E_2.t_s, ..., E_k.t_s\}\]  \hspace{1cm} (1)

\[\rho^- = \rho.t_e = \max\{E_1.t_e, E_2.t_e, ..., E_k.t_e\}\]  \hspace{1cm} (2)

Definition 8 (Pattern). A pattern \(\rho\) is represented as \(\rho = (X_1 \odot_1 X_2 \odot_2 \ldots \odot_{\omega\Delta\theta} X_k)\) where \(X_i\) is an event, \(X_i \in \{pE, iE, sE\}(1 \leq i \leq k)\), and \(\odot_i \in \{\cdot; \oslash; \odot; \parallel\}(1 \leq i \leq k-1)\).
is formulation operation. In case there are multiple occurrences of a \( X_t \) in a pattern, it is necessary to distinguish which two event boundaries (e.g. \( X_t^1 \) and \( X_t^2 \)) represent the same \( X_t \) occurrence. However, we assume that each event type has only one occurrence in a pattern and \( X_t^1 \) and \( X_t^2 \) are coming from the same occurrence of \( X_t \). Our language supports a hierarchy of complex patterns by feeding the output of one operator as an input to another. So a pattern operator not only connects events but also connects a number of pattern expressions to form a new expression. We now consider what it means that a pattern occurs in an event stream. Intuitively, the event types of the pattern need to have corresponding events in the stream such that the event types are the same and the operations between events of the pattern are respected and satisfied. The semantic of pattern expression, represented as \( p(t_1, t_2) \) or \( p(T) \) at a given time interval \( T = [t_s, t_e] \), \( p(t_s, t_e) \) is true if pattern \( p \) has an occurrence in time interval \( T \) (e.g. starts at \( t_s \) and ends at \( t_e \)). Frequency of a pattern is the number of occurrences of the pattern in an event stream. As shown in equations (1) and (2), a pattern has both start time and end time. So the temporal aspect of a pattern is a complete time interval. Two order relations can be defined on complete intervals:

Definition 9. A partial order \( \prec \) is defined as follows: \( \forall T, T' \in I, T < T' \Leftrightarrow t_s < t'_s \wedge t_e < t'_e \)

Definition 10. A total order relations \( < \) is defined as follows:

\( \forall T, T' \in I, T < T' \Leftrightarrow t_s < t'_s \lor (t_s = t'_s \land t_e < t'_e) \)

Selection Operation \( \sigma_P(p) \): This operator filters pattern expression on predicate \( P \), where \( P \) refers to event attributes contained in the pattern.

Sequence Operation \( (p; p_1; ...; p_k) \): This operator detects if pattern expression \( p_1 \) is followed by pattern expression \( p_k \) and so on. The operator specifies a particular order in which the patterns of interest should occur. Formally it defines as follows:

\( (p_1; p_2; ...; p_k) \Leftrightarrow \exists T_1 = [t_1_s, t_1_e], T_2 = [t_2_s, t_2_e], ..., T_k = [t_k_s, t_k_e] \)

such that \( T_1 < T_2 < \ldots < T_k, p_1(T_1) \land p_2(T_2) \land \ldots \land p_k(T_k) \). Considering temporal relations in Table 1, the sequence operation can have 4 sub-categories:

- \( (p_1; p_2) \Leftrightarrow (p_1; p_2^\delta) \equiv \exists T_1 = [t_1_s, t_1_e], T_2 = [t_2_s, t_2_e] \)

such that \( T_1 < T_2, p_1(T_1) \land p_2^\delta(T_2) \land t_1_s < t_2_s \)

- \( (p_1; p_2) \Leftrightarrow (p_1; p_2^\omega) \equiv \exists T_1 = [t_1_s, t_1_e], T_2 = [t_2_s, t_2_e], T_3 = [t_3_s, t_3_e] \)

such that \( T_1 < T_2 < T_3, p_1(T_1) \land p_2^\omega(T_2) \land p_3(T_3) \)

- \( (p_1; p_2) \Leftrightarrow (p_1; p_2^\delta) \equiv \exists T_1 = [t_1_s, t_1_e], T_2 = [t_2_s, t_2_e] \)

such that \( T_1 < T_2, p_1(T_1) \land p_2^\delta(T_2) \land t_1_s < t_2_s \)

Conditional Sequence Operation \( (p_1; \omega_{\Delta t}; p_2; \omega_{\Delta t} \ldots \omega_{\Delta t_k}; p_k) \): It detects if pattern expression \( p_1 \) is followed by pattern expression \( p_k \) within \( \Delta t \) time units. \( \Delta t \) is called the time lag or temporal restriction between two successive patterns. Formally it defines as follow:

\( (p_1; \omega_{\Delta t_1}; p_2; \omega_{\Delta t_2} \ldots \omega_{\Delta t_k}; p_k) \equiv \exists T_1 = [t_1_s, t_1_e], T_2 = [t_2_s, t_2_e], ..., T_k = [t_k_s, t_k_e] \)

such that \( T_1 < T_2 < \ldots < T_k, p_1(T_1) \land p_2(T_2) \land \ldots \land p_k(T_k) \). \( T_2 - T_1 \leq \Delta t_1 \land T_3 - T_2 \leq \Delta t_2 \land \ldots \land T_k - T_{k-1} \leq \Delta t_{k-1} \)

Concurrence Operation \( (p_1 \equiv p_2 \equiv \ldots \equiv p_k) \): Concurrence detects multiple patterns occur in parallel, and succeeds only if all patterns are detected. Unlike sequence, any order is allowed, and there has to be a non-empty overlap interval among the patterns.

\( (p_1 \equiv p_2 \equiv \ldots \equiv p_k) \equiv \exists T_1 = [t_1_s, t_1_e], T_2 = [t_2_s, t_2_e], ..., T_k = [t_k_s, t_k_e], p_1(T_1) \land p_2(T_2) \land \ldots \land p_k(T_k) \). 

Time (\( \omega_{\Delta t}; p \)): This operator requires a pattern \( p \) to occur within a certain time interval \( \Delta t = [\delta_1, \delta_2] \).

\( \omega_{\Delta t} \equiv \exists T = [t_s, t_e] \) such that \( p(T) \land \delta_1 \leq t_e \leq \delta_2 \).

Co-occurrence (\( CO_1; CO_2(\Delta t) \)): This operator computes if \( p_1 \) is co-occurring with \( p_1 \) within \( \Delta t \) time lag. Formally it defines as:

\[
CO_{p_1; p_2}(\Delta t) = \frac{\text{Count}(p_1; \omega_{\Delta t}; p_2)}{\text{Count}(p_1)}
\]

with \( CO_{p_1; p_2}(\Delta t) \in [0, 1] \) and \( \max[CO_{p_1; p_2}(\Delta t)] = 1 \) means that \( p_1 \) and \( p_2 \) are always co-occurring within \( \Delta t \) time lag, while values close to zero indicate that there is no co-occurrence within the specified time lag. To check for a significant time lag between \( p_1 \) and \( p_2 \), we define:

\[
\Delta t_{\text{max}} = \arg\max_{\Delta t} (CO_{p_1; p_2}(\Delta t))
\]

with \( \Delta t \in \{1, 2, \ldots, \lambda_{CO}\} \) where \( \lambda_{CO} \) is a design parameter indicating the maximum possible time lag between patterns.

2.4 Visualization Operators

Temporal data mining methods often identify frequently occurring patterns. However, the context in which these patterns occur is typically lost. Also complex patterns that contain both sequential and concurrent components cannot be formulated. So we define visualization operators that combines both mining and visualization techniques.

2D Co-occurrence Matrix: In image processing a co-occurrence matrix or co-occurrence distribution is a matrix that is defined over an image to be the distribution of co-occuring values at a given offset to measure the texture of the image. In our work, we use co-occurrence matrix in a different way. Consider two event streams \( ES^{(1)} \) and \( ES^{(2)} \) with \( N = [N^{(1)}] \) and \( M = [N^{(2)}] \) denote the number of event types in each event stream. Co-occurrence matrix \( C \) is an \( N \times M \) matrix where each cell \((i, j)\) is the co-occurrence value calculated from definition 3 for a pair of events \( E_i \) and \( E_j \):

\[
C_{\Delta t}(i, j) = CO_{E_i; E_j}(\Delta t) \quad \forall i = 1, \ldots, N; j = 1, \ldots, M
\]

In future, we will introduce a 3D Co-occurrence matrix operator that captures causal association between 3 event streams.
3. PROCESSING TECHNIQUE

The processing operators defined in the previous section are the basic building blocks of causal pattern recognition. These operators need to be compiled to a processing unit. We chose Finite State Automaton (FSA) as the underlying processing technique and each operator translates to its corresponding automaton (as shown in Table 3) to detect instances of a specific pattern in the input stream. The theory of finite-state automata is rich and finite-state automata techniques have been used in a wide range of domains, such as pattern matching, pattern recognition, speech processing, handwriting recognition, encryption algorithm, and data compression. So in our framework the pattern recognition component employs an extended FSA that supports a time model to address temporal restrictions that are needed in causal analysis. This automaton contains a finite number of states and state transitions. There are two types of states: ordinary states and time states. Ordinary state is analogous to states in traditional finite automata. It consumes an event from event stream, apply an EVALUATE() function and make a transition to the next state if the evaluation is successful. Time state on the other hand keeps track of time constraint requirement by applying a SET() function on boundaries of time lag \((\delta_1, \delta_2)\) that is going to be used in the evaluation function of the next ordinary state. Ordinary state is represented by an event type \(E_i \in \xi\) and it means that the FSA is waiting for \(E_i\) to be seen in the input event stream.

Definition 11 (Automaton): A finite-state automata (FSA) is a 5-tuple \((OS, TS, Ed, s_0, s_f)\), consisting of a finite set of ordinary states \((OS)\), time states \((TS)\), transitions between states \((Ed)\), a start state \(s_0 \in OS\), and a final or acceptance state \(s_f \in OS\).

The underlying processing technique is same for both hypothesis driven and data-driven analysis. However, hypothesis driven emphasis on an interactive reasoning using expert’s knowledge. First, analyst seeds a causal hypothesis from the input event stream. The strategy for counting occurrences of a pattern is straight forward. For a pattern, \(\rho\), an automaton \(FSA_{\rho}\) is initialized. During run time, \(E_i, t_j\), and \(E_i, t_e\) substitute with start and end time of an instance of event \(E_i\) from the input event stream. The strategy for counting occurrences of a pattern is straight forward. For a pattern, \(\rho\), an automaton \(FSA_{\rho}\) is initialized. The initialization process includes translating event types and temporal constraints to ordinary states and time states, and allocating a buffer for EVALUATE() and SET() functions within each state. As we read data from event stream, by considering the output of EVALUATE() and SET() functions within each state. Once it reaches the final state, an occurrence of the pattern is recognized and its frequency is increased by one. A fresh automaton is initiated for this pattern when an event corresponding to its first event appears again in the event stream.

In order to process complex patterns that are combination of multiple operators, we use a logical operator tree. The logical operator tree for a given causal pattern specifies the processing operators used and their order. Nodes in the operator tree represent configured operators (refer to the side panel in Fig. 2), and directed edges between nodes denote the the event streams flow. Analyst sequentially configure operators as a part of designing a causal hypothesis. This operator tree is then parsed, instantiated and converted to a runtime operator tree.

In data-driven analysis our goal is knowledge discovery, finding novel, interesting and interpretable patterns. As opposed to hypothesis driven, where a single complex pattern

---

Table 3: Corresponding automaton for each operation. (Note: For simplicity and without the loss of generality, operators are depicted between two events only)

<table>
<thead>
<tr>
<th>Operation Automaton</th>
<th>Automaton</th>
</tr>
</thead>
<tbody>
<tr>
<td>(\sigma_{\rho}(E_i))</td>
<td>(S_0 \xrightarrow{\text{EVALUATE}} E_1)</td>
</tr>
<tr>
<td>((E_i; E_j))</td>
<td>(S_0 \xrightarrow{\text{EVALUATE}} E_1 \xrightarrow{\text{EVALUATE}} E_2, E_3)</td>
</tr>
<tr>
<td>((E_i', E_j'))</td>
<td>(S_0 \xrightarrow{\text{EVALUATE}} E_1 \xrightarrow{\text{EVALUATE}} E_2, E_3)</td>
</tr>
<tr>
<td>((E_i \Delta \omega \Delta t; E_j))</td>
<td>(S_0 \xrightarrow{\text{EVALUATE}} E_1 \xrightarrow{\text{EVALUATE}} E_2, E_3)</td>
</tr>
<tr>
<td>((E_i \land E_j))</td>
<td>(S_0 \xrightarrow{\text{EVALUATE}} E_1 \xrightarrow{\text{EVALUATE}} E_2, E_3)</td>
</tr>
<tr>
<td>((\omega \Delta t; E_i))</td>
<td>(S_0 \xrightarrow{\text{EVALUATE}} E_1 \xrightarrow{\text{EVALUATE}} E_2, E_3)</td>
</tr>
</tbody>
</table>

Figure 5: The automaton corresponding to pattern \(\rho_1\) with 3 event components. It demonstrates 3 ordinary states, 2 time states, and EVALUATE() and SET() functions associated with each state.
Algorithm 1. Pseudocode for counting conditional sequential patterns

1: Input: Event Streams $E^0, E^1, \ldots, E^N$
2: A set of patterns $P$
3: $ES = merge (E^0, E^1, \ldots, E^N)$
4: $\Sigma = \Sigma^0 \cup \Sigma^1 \cup \ldots \cup \Sigma^N$
5: for all event types $E$ in $\Sigma$
6: \hspace{1em} waits($E$) = φ // Initialize waits($E$) index
7: end
8: for all patterns $p$ in $P$
9: \hspace{1em} Initialize an automaton $FSA_p = (OS_p, TS_p, Ed, so, se)$
10: \hspace{2em} Create a list of ordinary states $OS_p$
11: \hspace{2em} Create a list of time states $TS_p$
12: \hspace{2em} $\rho.freq = 0$
13: \hspace{2em} Add $(FSA_p, E)$ to waits(OSp[1].EventType)
14: end
15: for $i = 1$ to $n$ read the next event $e_i$ from $ES$
16: \hspace{1em} Let $E = e_i$ .type
17: for all $(FSA_p, x)$ pointed by waits($E$)
18: \hspace{2em} if $OS_p[x].EVALUATE() = TRUE$
19: \hspace{4em} Proceed to $TS_p[x]$ state
20: \hspace{2em} Execute $TS_p[x].SET()$
21: \hspace{2em} Let $x = x + 1$
22: \hspace{2em} Proceed to $OS_p[x]$ state
23: \hspace{2em} Remove $(FSA_p, x)$ from waits($E$)
24: \hspace{2em} if $OS_p[x].type = \text{final}$
25: \hspace{4em} $\rho.freq++$, $x = 1$
26: \hspace{2em} Add $(FSA_p, x)$ to waits(OSp[x].EventType)
27: \hspace{2em} else \hspace{1em} // temporal conditions are violated
28: \hspace{3em} $x = 1$ // start matching again
29: end
30: end
31: return $P$

was analyzed, in data-driven analysis a batch processing is performed on a collection of patterns. Then the result of the analysis is presented in visual graphics so analyst can explore data visually in order to extract information. For this batch analysis, our processing algorithm is a fast automaton based frequency counter that counts the frequency of a collection of patterns efficiently with only one pass through event streams. Algorithm 1 shows the pseudo-code of processing algorithm for conditional sequential patterns. We will present the algorithms for other types of patterns in our future work. For each pattern an automaton is initialized (line 9-12). A conditional sequential pattern $\rho$ has an automaton $FSA_p$, where $OS_p$ is a list of ordinary states $TS_p$ is a list of time states. $OS_p[x]$ is the $x^{th}$ ordinary state with $EventType$ and $StateType$ attributes and $EVALUATE()$ method. Similarly, $TS_p[x]$ is the $x^{th}$ time state with a $SET()$ method to set time constraint boundaries.

To efficiently access all automata, we index them using a waits(.) list. For each event type $E$, the automata that are waiting for $E$ are linked together in a list pointed by waits($E$). Giving an example from figure ??, $\rho_1$ is added to waits($E_5$) (line 13). Each element of this list is a pair $(FSA_p, x)$ indicating that FSA corresponding to pattern $\rho$ is in its $x^{th}$ ordinary state, waiting for $OS_p[x].EventType$ to be seen in the input stream. The idea of efficiently indexing automata through a waits(.) list was introduced in the windows-based frequency counting algorithm [10] with the difference that the list was used to manage up to $N$ automata per a pattern of size $N$. The algorithm we present here requires just one automaton per pattern. The main loop in the algorithm looks at each event, say $e_i$, in the input stream and makes necessary changes to the list of automata pointed by waits($e_i$.type). If an automaton reaches its final state, frequency of the pattern increases by one. At any state, if the validation function fails, automaton will not proceed with the current match and goes back to its start state. This algorithm is very efficient because it goes through the multi-event stream only once to count the frequency of all input patterns.

4. THE FRAMEWORK FOR COMBINING HYPOTHESIS DRIVEN AND DATA DRIVEN ANALYSIS

As mentioned, in an appropriately designed causal modeling platform, there should be a balance between data-driven and hypothesis-driven modeling techniques where the former helps in knowledge discovery and the latter keeps human in the loop for reasoning and causal modeling. In this section we explain how causal modeling framework can be used as an exploration environment through 3 experiments: 1) Visual discovery 2) Interactive pattern growth. 2) Hypothesis refinement.

The experiments are performed on synthetic data that is generated by embedding some causal patterns. At this point, we only used conditional sequential operators to generate sample patterns. Each pattern to be embedded in the dataset, consists of a specific ordered sequence of events and time lags between them. Data generation process is as follow: There is a timer that specifies the current time instant. Each time an event is generated, this timer specifies event’s start time. The duration of each event is picked from a normal distribution with a mean value $\mu$. After generating an event, timer is incremented with a small random integer. Each time the next event is to be generated, two decisions should be made. 1) Whether the event is going to have both start and end timestamps, or one of them might be missing randomly. This is controlled by the parameter $\alpha$, which is the probability that the next event has its both interval boundaries. If $\alpha = 1$ then generated event stream contains only events having complete intervals. 2) Whether the next event is to be generated randomly with uniform distribution over all event types or according to one of temporal patterns to be embedded. This is controlled by the parameter $\beta$, which is the probability that the next event is generated randomly. If $\beta = 1$ then data contains only noise with no sequential temporal patterns embedded. If it decides that the next event is to be from one of the patterns to be embedded, then we have a choice of continuing with a pattern that is already embedded partially or starting a new pattern.

4.1 Visual Discovery

Following the synthesized data generation approach dataset1 is generated with total number of events $n = 10^8$, number of event types $|\Sigma| = 22$, $\alpha = 0.3$, and $\beta = 0.2$. While generating the data, the following three patterns were injected:

\[
\begin{align*}
\rho_1 &= (E_3 \xrightarrow{15} E_6 \xrightarrow{20} E_{10}) \\
\rho_2 &= (E_8 \xrightarrow{30} E_{19}) \\
\rho_3 &= (E_{12} \xrightarrow{60} E_9 \xrightarrow{15} E_0)
\end{align*}
\]

Data generation with $\beta=0.2$ means that with 20% probability the next event is generated randomly and with 80%
probability pattern $\rho_i$ ($i = 1, 2, 3$) is continued or a new occurrence of $\rho_i$ is started. Suppose analyst doesn’t know what to look for and needs some preliminary knowledge about the data (unknown-unknown problem). Using 2D co-occurrence matrix operation and by changing temporal offset, multiple $2 \times 2$ co-occurrence matrices can be computed. Three of these matrices are demonstrated in Fig. 6. Such a visualization facilitates browsing co-occurrence characteristics in event streams, formulating hypothesis regarding those characteristics, and investigating potential causal relationships between events. Considering the visualization, analyst finds 5 causal patterns that seeds the following hypotheses: $1) \ h_1 = (E_3 \ : \omega_{15} \ E_6)$. $2) \ h_2 = (E_6 \ : \omega_{20} \ E_{10})$. $3) \ h_3 = (E_8 \ : \omega_{30} \ E_{19})$. $4) \ h_4 = (E_{13} \ : \omega_{60} \ E_9)$. $5) \ h_5 = (E_4 \ : \omega_{15} \ E_{20})$.

### 4.4 Interactive Pattern Growth

Analyst can use the knowledge discovered in the previous section and interactively formulate and analyze more complex patterns. This interaction cannot be displayed in the form of snapshots. So we explain a simple case of growing a pattern. Suppose analyst wants to find causal patterns of size 3 in the form of $E_i \rightarrow E_j \rightarrow E_k$. With 22 event types, there are $92 \times 22$ sequential patterns of size 3 (same event type cannot occur twice in a pattern). If we restrict the time lag $\Delta t \in \{15, 30, 60\}$, in case of conditional sequential patterns of size 3, the total number of patterns reaches 55440. However, by performing visual analytics in the previous step and formulating hypothesis $h_1$ to $h_5$, we only need to analyze 300 candidate patterns to find significant causal patterns of size 3. These candidate patterns are generated as follow:

- $E_3 \rightarrow E_6 \rightarrow E_i \forall i \in \{1, 2, \ldots, 22\}, i \neq 3, 6$
- $E_6 \rightarrow E_{10} \rightarrow E_i \forall i \in \{1, 2, \ldots, 22\}, i \neq 6, 10$
- $E_8 \rightarrow E_9 \rightarrow E_i \forall i \in \{1, 2, \ldots, 22\}, i \neq 8, 19$
- $E_{13} \rightarrow E_9 \rightarrow E_i \forall i \in \{1, 2, \ldots, 22\}, i \neq 13, 9$
- $E_9 \rightarrow E_{20} \rightarrow E_i \forall i \in \{1, 2, \ldots, 22\}, i \neq 9, 20$

So by incorporating experts knowledge from previous step, the search space for finding embedded patterns has significantly reduced. The frequency of these patterns are displayed graphically and analyst can easily recognize the significance of patterns.

### 4.3 Hypothesis Refinement

Suppose analyst has a vague hypothesis in her mind where $E_1$ is the effect and $E_2, E_3, E_4$ are presumed to be causes of $E_1$ in a specified order. Also the time lag between events are unknown but her domain expertise indicate that $\Delta t \in \{15, 30, 60\}$. Hypothesis refinement is beneficial in investigating which events are the true causes and what temporal precedence and exists between them. So, using sequence analysis, analyst can formulate a vague hypothesis as: $h = E_2; E_3; E_4; E_1$. For refining this hypothesis, system automatically generates the following causal patterns:

- $\rho_1 = E_2 \rightarrow E_1$, $\rho_2 = E_3 \rightarrow E_1$, $\rho_3 = E_4 \rightarrow E_1$, $\rho_4 = E_2 \rightarrow E_3$, $\rho_5 = E_2 \rightarrow E_4$, $\rho_6 = E_3 \rightarrow E_4 \rightarrow E_1$

Then the frequency of these patterns are computed and graphically displayed. Analyst can easily select the most significant ones and repeat the refinement if needed.

## 5. EXPERIMENT: ASTHMA RISK FACTOR MODELING

In this section we present the applicability of our causal modeling framework in extracting expressive causal rules in the context of a healthcare application, asthma management. Exposure to air pollution specifically Particular Matter (PM2.5) is linked with asthma exacerbation; however, the role played by meteorological factors such as temperature, humidity, rainfall, wind, and the complicated interrelations between these factors and asthma attacks are not well understood. Temporal structure and order relation between these environmental factors and their effect on asthma exacerbation comprise complex patterns called asthma risk factors. By extracting such causal patterns we create an asthma risk model that is important both for an asthmatic patient and public health.

From our causal modeling framework we use conditional sequential operator to formulate and recognize complex risk factors. First unified event streams are construct by abstracting trends of data streams using Symbolic Aggregate approxXimation (SAX) method [8]. Then algorithm 1 is applied on multiple event streams to recognize significant complex patterns that encode asthma risk factors. We use meteorological, pollution and tweet data collected for 18 months in Tokyo (as shown in Fig.1). The result of our experiment is presented as descriptive rules (32 causal patterns) that are easily understandable by an expert. Expert might perceive some insight within these rules or decide to seed a new hypothesis based on them and dig deeper to find even more interesting causal patterns.

Mizumuma et al. [11] shows that there is a strong relationship between real world events and what people tweet about. So, tweets can be leveraged for recognizing a situation in the real world. Once we have recognized a real world phenomenon from social messages then we can look for dig-
To define the number of topics beforehand. In order to build communities; each community is considered as one topic. Then, we use KeyGraph to build a set of communities. Further modifications, and scientific journal archives (e.g. ojphi.org).

5.1 Topic Modeling

KeyGraph [17], a network of keywords based on their co-occurrence in documents, is used to build a topic detector $E_v$. The KeyGraph is the set of non-overlap sub-graphs, call communities; each community is considered as one topic. The significance of this methodology is that there is no need to define the number of topics beforehand. In order to build $E_v$ for asthma topic, we first collect a set of documents relate to asthma, from the Internet (e.g. wikipedia, medical organizations), and scientific journal archives (e.g. ojphi.org).

Then, we use KeyGraph to build a set of communities. Finally, we pick a community that contains asthma keyword and collect all keywords that connect directly to this keyword as the $E_v$. We assume that an incoming tweet that contains at least one of the keyword in $E_v$ might be related to asthma.

5.2 Event Stream Modeling

First we need to extract meaningful events from data streams and generate corresponding event streams. Event model explained in section 2.2 is used to create an abstraction level on top of sensory data streams to mask the heterogeneity of the underlying data. For human sensory data, a burst detector [7] is applied to find bursting points on the topic histogram to generate asthma outbreaks event stream (effect events). Since physical sensory data is a time-series data stream, trend analysis plays a major role. It not only facilitates prediction of a new value of data within a certain interval of time, but also gives a holistic view of changing values. Therefore, instead of raw data, trend data is used to extract information. The Season-Trend Decomposition by Loess (STL) method, introduced in [8], is taken into account to decompose original time-series data into Trend, Seasonal, and Remainder streams. Then, SAX algorithm with alphabet size equal to 3 (a,b,c symbols) is applied to trend data in order to create SAX-code streams. Given the specifications of our application, we are interested in capturing not only state transitions (e.g. pollution level increases or decreases, point events) but also maintaining a specific state value for a duration of time (e.g. pollution level stays high, interval events). To do so, we define 6 event types for each stream. Table 4 shows a list of SAX codes and the corresponding event definition assigned to them. Using this encoding, each SAX-code trend data stream is converted to an event stream.

As explained earlier we are interested in finding the interrelation between multiple asthma triggers, mainly PM2.5 and meteorological factors. Complex risk factors are patterns characterizing by structural order and temporal constraints between events from multiple event streams. Going back to the definition of a pattern $(X_1 \circ \ldots \circ X_k)$, $X_k$ is burst in tweets (an indication of asthma outbreak), $X_1$ to $X_{k-1}$ are event types from air pollution and meteorological event streams, and $\circ_i$ (1 ≤ i ≤ k − 1) is considered to be $\omega_{\Delta t}$ where $\Delta t$ is 1 to 7 day time lag.

5.3 Causal Pattern Extraction

For extracting and evaluating asthma risk factor patterns we have used data collected for 18 months, from 2013-July-01 to 2014-Dec-30, in Tokyo city. Continuous hourly-average PM2.5, temperature, rainfall, and wind speed data are available for multiple sampling locations in Tokyo. Since asthma risk factors are quite different between different seasons, dataset is divided into 4 seasons: Spring (March - May), summer (June - August), autumn (September - November), and winter (December - February).

Fig. 7 demonstrates the most significant asthma risk factors of size 2. "PM2.5,increase,Asthma_outbreak" reads: once PM2.5 increases, an asthma outbreak happens within 3 days.)

![Graph showing asthma risk factor patterns](image)

Table 4: Definition of events assigned to each SAX-code

<table>
<thead>
<tr>
<th>Symbol</th>
<th>SAX-code</th>
<th>Event Definition</th>
</tr>
</thead>
<tbody>
<tr>
<td>Low Value Level=a</td>
<td>ab / bc</td>
<td>increase</td>
</tr>
<tr>
<td>Medium Value Level=b</td>
<td>ac</td>
<td>suddenly increase</td>
</tr>
<tr>
<td></td>
<td>ca</td>
<td>suddenly decrease</td>
</tr>
<tr>
<td>High Value Level=c</td>
<td>cb / ba</td>
<td>decrease</td>
</tr>
<tr>
<td></td>
<td>aaa</td>
<td>stay low</td>
</tr>
<tr>
<td></td>
<td>ccc</td>
<td>stay high</td>
</tr>
</tbody>
</table>

Figure 7: Frequency of asthma risk factors of size 2. "PM2.5, increase,Asthma_outbreak" reads: once PM2.5 increases, an asthma outbreak happens within 3 days.)
between two or three events as well as the time lag between them and asthma exacerbation (e.g., when rain decreases followed by PM2.5 increase within 4 days, then asthma outbreak is probable.) Our results suggest some interesting patterns: when PM2.5 increases followed by temperature stay high within 3 days, then asthma outbreak is probable; or when wind decreases followed by PM2.5 increases within 5 days, then asthma outbreak is probable. Recognizing the time lags between events is one of the most important contributions of our approach. To the best of our knowledge this aspect of correlation between different risk factors has never been studied systematically before in causal modeling for health care applications.

In addition to the above mentioned patterns, there are some patterns that might seem counter intuitive. For instance: When rain increases followed by PM2.5 stay low within 4 days then an asthma outbreak is probable. Such cases are either noisy patterns or there might be some justifications when considering other climate/environmental effect and shall be investigated by experts.

6. CONCLUSIONS

In this paper we presented a causal modeling technique that facilitates deep computational understanding and model building in different domains using traditional as well as emergent new multimedia big data streams. We argued that there should be a balance between data-driven and hypothesis-driven modeling techniques since they complement each other. A formal causal pattern language consists of processing and visualization operators was defined and an extended FSA was applied as the underlying processing technique. By using this framework, on one hand analyst can benefit from visual operators for knowledge discovery and on the other hand she can perform separate analysis for each causal assumption and interactively refine the generated hypothesis. Some preliminary results using conditional sequence operator for asthma risk factor recognition is presented. In future we apply this framework to build a comprehensive user model (objective self [5]) from personal data streams.

7. REFERENCES

[18] E. Winarko and J. F. Roddick. Armada—an algorithm for discovering richer relative temporal association