eVM: An Event Virtual Machine Framework

Elio Mansour, Richard Chbeir, Philippe Arnould

eVM: an event Virtual Machine Framework

Elio MANSOUR, Richard CHBEIR, and Philippe ARNOULD

UNIV PAU & PAYS ADOUR / E2S-UPPA, LIUPPA, EA3000, FRANCE
{elio.mansour, richard.chbeir, philippe.arnould}@univ-pau.fr

Abstract. Information and communication technology (ICT) is impacting our daily lives more than ever before. Many existing applications guide users in their daily activities (e.g., navigation through traffic, health monitoring, managing home comfort, socializing with others). Although these applications are different in terms of purpose and application domain, they all detect events and propose actions and decision making aid to users. However, there is no usage of a common backbone for event detection that can be instantiated, re-used, and reconfigured in different use cases. In this paper, we propose eVM, a generic event Virtual Machine able to detect events in different contexts while allowing domain experts to model and define the targeted events prior to detection. eVM simultaneously considers the various features of the defined events (e.g., temporal, geographical), and uses the latter to detect different feature-centric events (e.g., time-centric, location-centric). eVM is based on different components (an event query language, a query compiler, an event detection core, etc.), but mainly the event detection modules are detailed here. We show that eVM is re-usable in different contexts and that the performance of our prototype is quasi-linear in most cases. Our experimental results showed that the detection accuracy is improved when, besides spatio-temporal information, other features are considered.

Keywords: Event Detection, Semantic Clustering, Formal Concept Analysis.

1 Introduction

During the last decades, the world has witnessed important changes. Most notably, information and communication technology (ICT) has become a major part of our lives. Recent research efforts have brought forward massive advances in the field of data management, thus easing the way for ICT to be tightly integrated in various application domains (e.g., social, medical, home/building management, energy management, navigation systems, industry and manufacturing). As a result, we now have numerous applications that help users in their everyday tasks. To give a few examples, consider social applications that allow users to socialize, comment on social issues and gatherings, and share multimedia data (e.g., photos, videos) taken by different users, during life events [27] through a simple interface. Social media have evolved greatly over the past decade. As a result, more users are now connected to collaborative and information sharing

platforms (e.g., Facebook, Google+) and other social sharing applications (e.g., Iphotos[1]). The medical domain is also witnessing the evolution of ICT. Nowadays, patients use personal sensors to monitor their health. Several studies [17, 20, 41] detail the usage of wearable sensors and data mining techniques in order to monitor a patient's heart condition, or gait analysis (i.e., a person's manner of walking). Buildings and homes are also evolving. With the integration of sensor networks and advanced data processing techniques, we are currently in the era of automated homes and smart buildings [6, 43]. In contrast with traditional buildings/homes, these dynamic systems provide a more comfortable [22], sophisticated [40], and healthy [46] environment for occupants. Other works [1] focus more on energy management in these infrastructures. In this regard, the integration of data mining and sensor networks reduced energy costs for building/home owners, and more importantly diminished the environmental impact of the aforementioned infrastructures. ICT is also impacting the way we travel, drive, and move around in our cities. Navigation and driving guidance systems [44] help users on the roads everyday. Using geo-localization, sensor networks, and modern telecommunication, drivers can now avoid traffic jams/congestion, and report occurring incidents. This is making roads safer and more accessible for everyone. The integration of ICT even reached the industrial domain [23]. Currently, factory managers are using machine monitoring applications that allow them to better schedule machine maintenance, thus preventing faults and breakdowns.

All of the aforementioned works are different (in terms of application domains, purposes, and objectives). However, they share the need to detect important events. To achieve this, many works [8, 9, 32, 36, 37] have evolved around the organization of shared data on platforms and applications using clustering techniques. Some social-based studies [8, 32, 35, 37] are based on metadata (e.g., Facebook organizes multimedia content based on publishing timestamps, Iphotos combines photo creation timestamps and locations to organize a user's photo library). Others [26, 28, 31] use the visual attributes of shared objects (e.g., textures, colors) coupled with metadata in order to detect several events. Medical/health monitoring applications [17, 20, 41] detect health-related events for a specific patient. In the case of urgent/life threatening events, event detection allows an efficient and timely intervention from the patient's doctor. Otherwise, information regarding the detected events can be used for deep analysis of the medical issue, testing, diagnosis, and better decision making. Home/building and energy management works [1, 22, 40, 46] also detect events. For example, detecting if a person entered a room in order to automatically turn on the lights, or detecting if the building is empty to turn off the heating and reduce energy consumption. Furthermore, driving guidance systems [44] need to detect events such as accidents, traffic jams, and road maintenance in order to steer drivers away from these events'locations. Finally, machinery monitoring applications [23] automatically detect events that could lead to machine breakdowns and faults in order to optimize maintenance scheduling.

[1] http://www.apple.com/ios/photos

All the aforementioned works share common principles: (i) they need to detect a set of events; and (ii) they need to extract data and apply at least one data mining technique (e.g., clustering, classification). The main differences are two fold: 1) the targeted events and therefore their features, and 2) the choice of data mining/event detection technique.

Even-though all these works detect events, there is no usage of a common backbone for event detection that can be instantiated in different contexts/application domains. This is restrictive and costly since existing solutions suffer of the following issues: (i) the absence of an evolutionary approach capable of coping with needs that change over time; (ii) the absence of extensibility regarding the integration of new plug ins/complementary systems to an existing event detection approach; (iii) the difficulty of integrating different event-related modifications in the development; (iv) the impossibility of reusing the same framework to detect other events in various domains/contexts; and (v) the lack of expert input, i.e., providing a module where one can provide his own input on how to define the corresponding events prior to detection. In addition, sometimes data shared/published is heterogeneous, and this generates some technical aspects to be considered: multi-modality (the ability to consider various features and datatypes at once in the processing), incremental processing (allowing a continuous integration of new data (e.g., new sensor data, new photos shared/published on different dates/times) in the set of already processed data), and multi-source processing (considering various data providers at once). Thus, there is a need to design a generic event detection approach, considering expert input, and event features, to provide a more reusable approach.

To answer this need, we propose eVM: a re-usable architecture for automatic and generic detection of "feature-centric" events. Our framework is composed of several main components: 1) An event Query Language (eQL): one can use the eQL to create, update, delete, select, or insert datatypes and event features, thus defining the targeted events. In addition, one can define several key event features. Based on the latter, our approach detects the corresponding feature-centric events. This allows the framework to be generic and re-usable in different event detection contexts/domains while allowing domain experts to provide their input; 2) An easy to integrate API containing an Event Query Compiler for eQL query processing and an Event Detection core. This makes eVM evolutionary, extensible, and easy to integrate with other modules/systems and programing languages; and 3) a storage space where various repositories are available for the storage of various event related data (e.g., data objects, event definitions, detected feature-centric events).

eVM's clustering technique simultaneously considers the various features, objects shared by different sources (several data producers) on different dates/times, as well as data from one source (i.e., to organize a source's data library). eVM is based on an adaptation of FCA (Formal Concept Analysis) [16, 42], a backbone that provides a multi-modal, incremental, and multi-source clustering technique that handles high dimensional data, and requires low human intervention.

In order to validate our approach, we implemented eVM as a cross-platform mo-

bile application in order to evaluate the approach in different real case scenarios. Our experimental results, on ReSEED Dataset [34] and another simulated one, show that the event detection accuracy is improved when additional features (i.e., other than time and geo-location) are taken into consideration. In addition, our performance results show quasi-linear behavior in most cases.
The rest of the paper is organized as follows. Section 2 reviews event detection works. Section 3 introduces FCA and defines the eVM approach. The implementation and evaluation are discussed in Section 4. Section 5 reviews clustering techniques. Finally, Section 6 concludes and highlights future perspectives.

2 Related Work

In this section, we review event definitions, types, and features before detailing event detection works in different areas (e.g., social event detection, medical event detection, sensor event detection). Since, in most cases, events are detected based on incoming raw data, without having any prior knowledge on the occurring events, main approaches [8, 28, 31, 32, 37] use unsupervised clustering techniques. Since there are no commonly adopted criteria, we propose the following set of criteria to compare the referenced works:

1. *Re-Usability*: This criterion examines the possibility (*yes* or *no*) of using the same approach as an event detection backbone for different targeted events in different contexts/domains (cf. Section 1 - limitation (iv)).
2. *Domain Specific Expertise*: This criterion measures the possibility (*yes* or *no*) of taking into account input from domain experts in the event definition process (cf. Section 1 - limitation (v)).
3. *Evolution*: This criterion examines (*yes* or *no*) if an event detection approach can evolve and adapt to the changing event definition/detection needs over time (cf. Section 1 - limitation (i)).
4. *Extensibility*: This criterion measures (*yes* or *no*) the capability of integrating new plug-ins/external modules in an existent event detection approach (cf. Section 1 - limitation (ii)).
5. *Ease of Integration*: This criterion denotes (*yes* or *no*) an approach's capability of integrating event-related modifications in the development (cf. Section 1 - limitation (iii)).

In addition to the aforementioned criteria, we also consider other technical requirements/criteria such as:

6. *Multi-modality*: This criterion states (*yes* or *no*) if multiple event features having different datatypes are considered (e.g., *social*, *topics* (textual information), *sensor-related* (scalar and multimedia information)) in addition to time (datetime) and locations (GPS coordinates or textual location description) for improved event detection.
7. *Multi-source*: This criterion indicates (*yes* or *no*) if multiple data sources (various publishers, data producers) are considered. This is important since various sources can provide valuable event related data.

8. *Incremental (continuous) processing*: This criterion considers the possibility (*yes* or *no*) of processing incoming data without having to repeat the entire processing, because data producers could share event related data on different dates/times.
9. *Level of human intervention*: This criterion measures (*high*, *moderate*, or *low*) how frequently users participate in the event detection process; since huge amounts of data are shared, it is important that user interventions become less frequent; we consider low intervention if users provide data input and initial configuration; moderate if users also intervene in result correction/optimization; and high intervention when users participate in the whole process.

In the following, we begin by defining events, before detailing research works from various domains in which event detection has had noticeable impact (e.g., social event detection, sensor event detection).

2.1 Basic Definition of an Event

In the literature, many works [2, 3] define events as a happening that takes place at a particular time and location. Thus, emphasizing the importance of two main event features: (i) temporal; and (ii) spatial. All events are associated with these two features, since they answer the most common inquiries i.e., where and when. Nonetheless, additional event features are useful to describe the context and semantics of an event (e.g., social, political, medical). The additional features differ from an event to another.
Events are categorized into different types, regardless of their contexts: (i) atomic or primitive events are the simplest events that can occur in a system. They cannot be decomposed into any smaller entity; (ii) composite or complex events are high level derived events, and are defined by combining constituent events. The latter can be atomic, or/and composite [2]. In our proposal, we currently consider atomic events, nonetheless the framework is re-usable and extensible, and can easily integrate a module for event composition that allows the detection of composite events.

2.2 Event Detection Applications

As mentioned before, event detection covers a large spectrum of application domains. From social event detection and sharing applications to environmental monitoring (e.g., detecting fire hazards in forests, level of air pollution in a city), energy management (e.g., detecting energy wastes in smart buildings), industrial processes (e.g., detecting events that disrupt production flow in a factory, detecting faults and machine maintenance issues), and medical event detection (e.g., monitoring a patient's heart condition) in various sensor networks. In all the aforementioned works, event detection requires a sensing (data collection) phase. During this phase, data is collected either through social means (shared images, videos, and posts on social media platforms) or physical equipments

(e.g., sensor observations produced by deployed sensors). Therefore, we discuss event detection works by their respective categories: (i) Social-based; and (ii) Equipment-based works.

Social-based Works: In the literature, several Social Event Detection (SED) approaches have emerged for detecting events. SED approaches can be grouped into two categories: approaches that rely on the metadata of shared objects (e.g., photos, videos, tweets) [8, 32, 35, 37], denoted metadata-based, and approaches that rely on visual attributes (e.g., colors, shapes) and metadata [13, 14, 26, 28, 31], denoted hybrid.

- Metadata-based approaches: In [32], the authors aim to detect social events based on image metadata, using temporal, geo-location, and photo-creator information. They perform a multi-level clustering for these features. In [8], the authors use time and GPS data to cluster photos into events using the mean-shift algorithm. First, the authors find baseline clusters based on time, then GPS location attributes are integrated. In [35], the authors use time and location information from twitter feeds to detect various events (e.g., earthquakes). In [37], the authors rely on textual tags such as time, geo-location, image title, descriptions, and user supplied tags to cluster photos into events, thus detecting soccer matches that happened in Madrid. These approaches need moderate human intervention. However, they are not incremental nor extensible. Metadata is also used by stand-alone applications for photo management to detect social events in a user's multimedia library. These applications require no human intervention, they automatically cluster objects found in a library. However, they do not consider other event features (e.g., social, topic), nor other photo sources (photos taken by other participants/collaborators). They mainly focus on time and location, photos taken at the same day and place of an event are merged with the event.
- Hybrid approaches: Many hybrid approaches rely on both visual and metadata attributes. In [28] and [31], the authors combine visual object attributes with temporal information, geo-locations, and user-supplied tags for their clustering procedures. Visual and tag similarity graphs are combined in [28] for the clustering. While in [31] the authors divide the geographical map of the world into square tiles and then extract the photos of each tile using geo-location metadata. They later use other metadata combined with visual features to detect objects and events. In [14, 26], the authors combine temporal metadata with visual attributes for annotation and event clustering purposes. In [13], the author relies more on temporal metadata than visual attributes for correct event detection, since he considers that photos/videos associated with one event are often not visually similar. Hybrid approaches consider different types of object attributes (e.g., visual, temporal, geo-locations). However, regrouping visually similar objects does not imply that they belong to the same event. Therefore, metadata is required to boost the accuracy of such approaches. Since these methods process visual attributes (e.g., through photo/video processing techniques), they end

up having a higher processing cost than the approaches that only process metadata. Some approaches require more human intervention, because they prompt the user to correct/optimize the results.

Equipment-based Works: With the evolution of sensor technology [11], sensor networks have been highly used for various purposes (e.g., personal sensing, social sensing, environmental monitoring), especially since the sensor data modeling aspects have been widely covered recently (e.g., through ontologies such as the Semantic Sensor Network SSN [12]). Sensors produce observations related to certain properties (e.g., temperature, movement, humidity). Regarding event detection in sensor networks, events are usually composed of sensor features (e.g., temperature) alongside spatio-temporal features since every sensor observation is mapped to an instant in time and a specific location. Many works [2, 25] agree that sensor observations are considered atomic events (e.g., temperature rise event), therefore works regarding sensor data fusion [4] could target the composite events. In the following, we detail some event detection works in sensor networks, based on the application domains.

- Medical Event Detection: In [17], the authors use lightweight, wearable, and durable sensors to monitor patients'gait (i.e., the manner of walking). People who suffer from strokes or spinal cord injuries, tend to have abnormal gaits. During medical treatment, it is beneficial to detect gait events when they occur (e.g., initial foot contact). The authors propose two different ways for detecting such events, one using accelerometer data, and another using foot switch data (i.e., data from pressure/force sensor). In both cases, the event features are spatio-temporal, and sensor-related (i.e., accelerometer, pressure, force). The authors test both cases on normal, slow, and altered walking subjects and achieve near real time accurate detection of abnormal gait events. In [20], the authors declare that a variety of measurements are required for gait analysis (e.g., stride, step lengths, cadence, gait velocity). In order to acquire such measurements, the system needs to know when and where each foot leaves and touches the ground again. Therefore, the authors take interest in detecting the following events: (i) foot end contact (EC); and (ii) initial contact (IC). Thus, the authors propose an approach for IC and EC event detection using linear accelerometers and angular velocity transducers. They then use the event detection results to analyze gait patterns of healthy and injured individuals. In [41], the authors propose a wireless smart sensor for heart monitoring. The aim is to detect life threatening events such as cardiac arrhythmia for patients with heart related issues. The sensor monitors heart rate and ECG signals to detect the aforementioned events in real time.
- Environmental Event Detection: In environmental monitoring scenarios, the sensor network contains more nodes (compared to personal medical sensing), and the spatio-temporal data acquisition intervals are wider. For example, to detect high air pollution events in a city, a huge number of air quality sensors should be deployed. In [15], the authors detect wildfire events in the wild

by collecting sensor data such as temperature, relative humidity, and barometric pressure. In addition, they integrate spatial features by using a GPS unit in order to localize the detected events. Information is communicated using a wireless sensor network. In [45], the authors propose an approach for real-time forest fire detection. They rely on spatio-temporal information and fire event context features such as relative humidity, temperature, smoke, and wind speed. They produce a report of abnormal atomic events (e.g., high temperature, smoke rising), and a real time forest fire danger rate from the collected data. Then they use a neural network to detect the fire events. In [24], the authors use crowd-sensing in order to detect and monitor air quality related events in a city. In addition to time and geo-localization, air quality events share features that are related to the context of air quality (e.g., carbon monoxide (CO), air pressure, nitrogen dioxide (NO2), and temperature). They also develop an android mobile phone application to display results to end users.

Table 1: Event Detection Works Comparison

Criterion	**Event Detection Category**				
	Social-based Works			Equipment-based Works	
	Metadata		Hybrid	Medical	Environmental
	[8, 32, 35, 37]	Stand-alone Applications	[13, 14, 26, 28, 31]	[17, 20, 41]	[15, 24, 45]
Re-Usability	No	No	No	No	No
Domain Specific Expertise	No	No	No	No	No
Evolution	No	Partially[2]	No	No	No
Extensibility	No	Partially[2]	No	No	No
Ease of Integration	No	No	No	No	No
Multi-Modality	No	Yes	Partially[2]	Yes	Yes
Multi-Source	Partially[2]	Partially[2]	Partially[2]	No	No
Incremental Processing	No	Partially[2]	No	Yes	Yes
Level of Human Intervention	Low - Moderate	Low	Moderate - High	Moderate	Moderate

Discussion: Table 1 summarizes the evaluation of event detection approaches based on the aforementioned criteria. Concerning Social-based works, metadata-based approaches [8, 32, 37] need low to moderate human intervention and provide good event detection accuracy, since metadata describes data related to the events (e.g., dates, locations, tags). However, these works lack the incremental processing needed to match the flow of publishing/sharing. Recently, incremental processing was integrated in some works (e.g., iPhotos). Hybrid methods [13, 14, 26, 28, 31] are costly computation-wise and require human intervention thus making continuous processing hard to implement [29]. In contrast, these methods offer more event features by combining visual attributes with metadata to improve accuracy. Finally, the two categories of works are not re-usable in different contexts nor fully consider the various features and modalities (datatypes) in event detection. They also do not consider domain expert input when defining the targeted events. Concerning Equipment-based works, we split these approaches based on their application domains. Nonetheless, they share two common characteristics: (i) they rely on a data acquisition networks (constituted of one or more

[2] Partially states that not all approaches of a category comply with the criterion.

sensors); and (ii) although they target different events, spatio-temporal event features are used by all methods. When considering the latter features, the chosen granularity can vary based on the application (e.g., for the spatial feature: we consider cities for environmental monitoring and the specific indoor location of a patient in a fall detection system). What differentiates the event definition from one approach to another are the specific (context related) features (e.g., temperature, movement, humidity). No current approach allows domain experts to contribute in event definition, i.e., the same event (e.g., abnormal gait) has variant definitions in different approaches. These works are not re-usable in different domains and contexts. When considering the other criteria we find that the level of human intervention varies from an approach to another. Finally even though these approaches are incremental, in most cases they are not extensible.

3 The event Virtual Machine (eVM) approach

In this section, we detail the eVM framework. Firstly, we provide an overview of our proposal. Then, we focus more on the clustering technique used in our event detection process. Finally, we detail our query language, modules and algorithm.

3.1 Approach overview

As previously discussed in section 2.1, existing approaches heavily rely on time and location when defining the targeted events. In order to tackle specific events, additional context related information (e.g., social, medical) are added. Nonetheless, existing works, such as [17, 32, 35], provide a static event definition that does not allow modifications (adding, removing event features). Since event definition and detection needs change over time, or from one context/domain to another, we provide a generic and extensible event definition in eVM. We model events as $2D^+$ spaces (having at least two dimensions: temporal and spatial). Depending on the targeted events, one can specify/create additional dimensions, representing the contextual event features such as *topics* (based on *tags* or *annotations*), *temperatures* (based on *sensor readings*) and various levels of granularities for each feature (e.g., year-month-week-day-hour for time, country-region-city-street for location). Micro granularities can also be considered depending on the application purpose (e.g., time: minutes-seconds for short events, location: human body-part of a body for medical applications). This way of defining events is dynamic, allows domain expert input, and is extensible. Nonetheless, it does not consider yet digital events, where the location dimension should be handled differently (e.g., server attacks that could happen on multiple nodes in the same time.)

Depending on the event detection needs, domain experts can create, insert, update and delete datatypes, event features, granularities, and event definitions using our proposed event Query Language (eQL). Every time a eQL query is submitted, a common Event Query Compiler (cf. Fig. 1) executes the submitted

query, thus creating an instance of event detection. Finally, using the defined instance, a common Event Detector mechanism is triggered. This process queries event related data, stored in specific repositories (found in the storage space), and starts detecting events based on the provided event model (definition). The Event Detector integrates FCA as the backbone clustering technique, to provide a multi-modal, dynamic, and incremental approach (the API components are later detailed in Fig.3). This makes the eVM framework re-usable in different contexts, evolutionary, and easy to integrate with any API friendly programing language. In this paper, we only detail the Event Detection part of this framework.

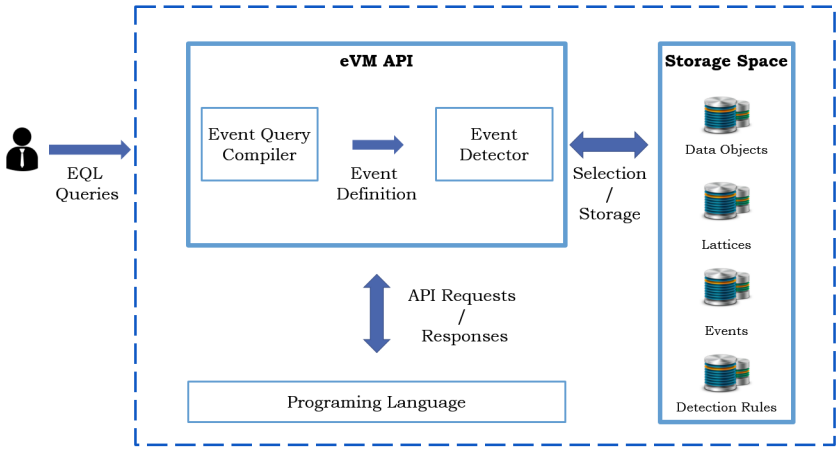

Fig. 1: eVM Overview

3.2 FCA Preliminaries & Definitions

After studying various clustering techniques [5, 7, 19, 33], we chose Formal Concept Analysis (FCA) [16, 42] as the backbone for the event detection process of our approach. FCA is incremental and multi-modal (criteria 8 and 6). It examines data through object/attribute relationships, extracts formal concepts and orders the latter hierarchically in a Concept Lattice which is generated through a four step process [10]:

Step 1: Defining a **Formal Context** (Def. 1) from the input data, based on object/attribute relations represented in a cross-table.

Definition 1 *A **Formal Context**: is a triplet $\langle X, Y, I \rangle$ where:*

- *X is a non-empty set of objects*

– *Y is a non-empty set of attributes*
– *I is a binary relation between X and Y mapping objects from X to attributes from Y, i.e., $I \subseteq X \times Y$.* ■

Table 2 shows an example, where photos are objects and photo attributes (locations, photo creator names, and dates) are attributes. The cross-joins represent the mapping of photos to their respective photo attributes, e.g., photo 1 *was taken in Biarritz by John on 17/08/2016.*

Table 2: Formal Context example

		Names				**Locations**			**Dates**		
		John	Patrick	Dana	Ellen	Biarritz	Munich	Paris	17/08/2016	12/12/2012	02/02/2016
Photos	1	x				x			x		
	2	x				x			x		
	3		x				x			x	
	4			x			x			x	
	5				x			x			x

Step 2: Adopting **Concept Forming Operators** to extract **Formal Concepts** (Def. 2). FCA has two concept forming operators:

– $\uparrow$: $2^X \rightarrow 2^Y$ (Operator mapping objects to attributes)
– $\downarrow$: $2^Y \rightarrow 2^X$ (Operator mapping attributes to objects).

For example, from the cross-table shown in Table 2, we have $\{3\}^{\uparrow} = \{$Patrick, Munich, 12/12/2012$\}$ and $\{02/02/2016\}^{\downarrow} = \{5\}$.

Definition 2 *A **Formal Concept** in $\langle X, Y, I \rangle$ is a pair $\langle A_i, B_i \rangle$ of $A_i \subseteq X$ and $B_i \subseteq Y$ such that: $A_i^{\uparrow} = B_i \wedge B_i^{\downarrow} = A_i$.* ■

Consider the set of photos A_1=$\{1, 2\}$ and the set of attributes $B_1 = \{$John, Biarritz, 17/08/2016$\}$. $A_1^{\uparrow} = \{$John, Biarritz, 17/08/2016$\}$ and $B_1^{\downarrow} = \{1, 2\}$. Thus, since $A_1^{\uparrow} = B_1$ and $B_1^{\downarrow} = A_1$, the pair $\langle A_1, B_1 \rangle$ is a Formal Concept.

Step 3: Extracting a **Subconcept/Superconcept Ordering** relation for **Formal Concept** (cf. Def. 2) ordering by defining the most general concept and the most specific concept for each pair. The ordering relation is denoted $\leq$.
For example, from Table 2, let $A_1 = \{3\}$, B_1 =$\{$Patrick, Munich, 12/12/2012$\}$, A_2=$\{3, 4\}$, and B_2 =$\{$Munich, 12/12/2012$\}$. According to Def. 2, $\langle A_1, B_1 \rangle$ and $\langle A_2, B_2 \rangle$ are formal concepts. In addition, $A_1 \subseteq A_2$ therefore, $\langle A_1, B_1 \rangle \leqslant \langle A_2, B_2 \rangle$. This means that formal concept $\langle A_1, B_1 \rangle$ is a subconcept of formal concept $\langle A_2, B_2 \rangle$ (which is the superconcept).

Step 4: Generating the **Concept Lattice**, which represents the concepts from the most general one (top) to the most specific (bottom). The lattice is defined as the ordered set of all formal concepts extracted from the data (based

on $\leq$). A **Concept Lattice** denoted by $\beta(X, Y, I)_{\leqslant}$ is the set of all formal concepts of $\langle X, Y, I\rangle$ ordered by the subconcept/superconcept ordering relation $\leqslant$, i.e.,

$$\beta(X, Y, I) = \{\langle A, B\rangle \in 2^X \times 2^Y | A^{\uparrow} = B, B^{\downarrow} = A\}$$

$\beta(X, Y, I)_{\leqslant}$ associated with a subconcept/superconcept ordering relation is called a concept (Galois) lattice. ■

For the example shown in Table 2, Fig. 2 illustrates the Concept Lattice. The top node is the concept regrouping all objects having no attributes in common. As we go down in the hierarchy, we notice that concepts have less objects and more shared attributes (a logic OR and AND are applied to objects and attributes respectively when scrolling down towards the bottom node). The bottom node is the most specific, thus regrouping all attributes having zero objects in common. The next section formally describes our eVM framework and how these FCA four steps are integrated and adapted for the clustering of data objects.

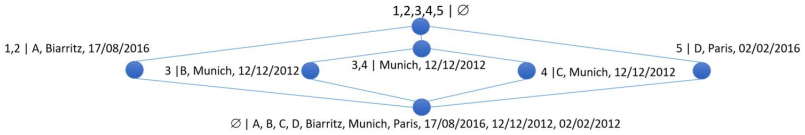

Fig. 2: The Concept/ Galois Lattice

3.3 The eVM Framework

In order to organize a set of event-related data objects according to feature-centric events, the eVM's API mechanism is split into four main steps: (i) Event definition & data pre-processing (executed by the Event Query Parser, Event Query Executor, and Pre-Processor modules); (ii) Attribute extraction (executed by the Attribute Extractor module); (iii) lattice construction (executed by the Event Candidates Lattice Builder module); and (iv) event detection (carried out by the Feature-Centric Event Detector and Rule Selector modules). In the following, we detail each processing step and module.

Event definition & data pre-processing: Through the Event Query Compiler, one uses the SQL-like event Query Language (eQL) to define the targeted events. The parser checks the syntax of the submitted queries. Then, the query statements are executed using the Event Query Executor module. Query statements are categorized into two groups: (i) Event Definition Statements (queries used to CREATE, UPDATE, INSERT, and DELETE datatypes, features, dimensions, and event spaces); and (ii) Event Selection Statements (queries used to select data or events from repositories. In this paper, we only detail main

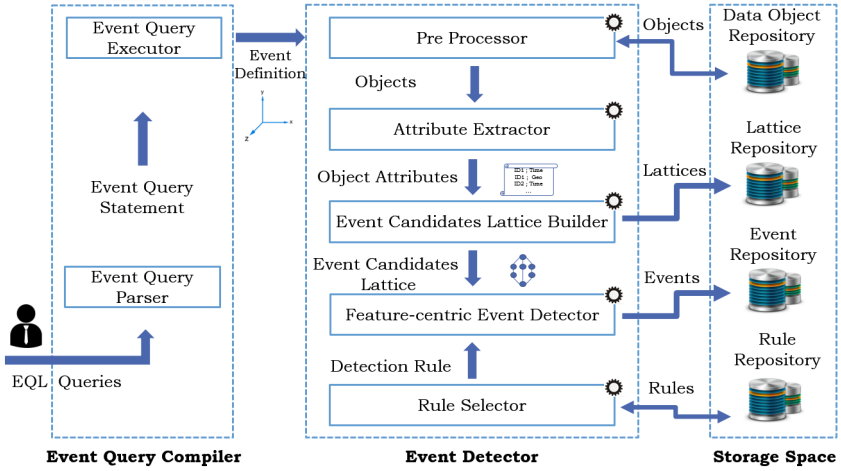

Fig. 3: eVM API Components

query statements. Details about eQL will be provided in a dedicated study. When defining the targeted events, several queries are to be considered such as event feature (`FEATURE`), attribute datatype (`ADT`), dimension (`DIMENSION`), and event space (`eSPACE`) creation. The following syntax shows how to do that: This query creates an event feature:

```
CREATE FEATURE <feature_id> (
        [label =] <value>,
        [G= ] {<value>},
        [interval =] '1' | '0',
        [gran =]  <function>,
);
```

Where `feature_id` is the unique identifier of the feature, `label` is the feature's label, `G` is a set of granularities associated with the feature, `interval` is a boolean indicating if the feature is generated as an interval (true), or not (false), and `gran` is a function that converts any granularity value to another (related to the same feature). For example, the following describes five (fire) event features each having a label, a set of granularities, an indication about interval construction, and a conversion function:

- $Time_f : \langle 'Time', \{Year, Month, Week, Day, Hour\}, 1, Convert_{Time} \rangle$
- $Geo_f : \langle 'Geo', \{Country, Region, City, Street\}, 0, Convert_{Geo} \rangle$
- $Temp_f : \langle 'Temp', \{value, setofvalues, mean, max, min\}, 1, Convert_{Temp} \rangle$
- $CO2_f : \langle 'CO_2', \{value, setofvalues, mean, max, min\}, 1, Convert_{CO2} \rangle$
- $Smoke_f : \langle 'Smoke', \{singlevalue, setofvalues\}, 0, Convert_{Smoke} \rangle$

For example, Time, Geo, Temp (temperature), CO2, and Smoke features could be used to define a fire event that can be detected from sensor data in a sensor network.
To create an attribute datatype, the syntax of eQL is as follows:

```
CREATE ADT <adt_id> (
[label =] <value>,
[t =] 'Integer'|'Float'|'Boolean'|'Date'|'Time'|
'Date Time'|'Character'|'String',
[range =] {<value>},
[dist =]  <function>,
[f =] <value>,
);
```

Where `adt_id` is the unique identifier of the attribute datatype, `label` is the attribute datatype's label, `t` denotes the primitive data type of the attribute, `range` is the domain of the attribute values, `dist` is the function that returns the distance between any two values of the same attribute datatype, and `f` is the event feature mapped to the attribute data type. To continue with the fire event example, the following describes five attribute data types each having a label, a primitive datatype, a range, a distance function (e.g., time difference for temporal attributes, spatial distance for geographical attributes, temperature, CO2, and Smoke differences between various sensor readings for instance), and an associated event feature:

- $Time_{adt} : \langle 'TimeAttribute', Date, Any, TimeDifference, Time_{feature} \rangle$
- $Geo_{adt} : \langle 'GeoAttribute', String, Any, SpatialDistance, Geo_{feature} \rangle$
- $Temp_{adt} : \langle 'TempAttribute', Float, Any, TempDifference, Temp_f \rangle$
- $CO2_{adt} : \langle 'CO2Attribute', Float, Any, CO2Difference, CO2_f \rangle$
- $Smoke_{adt} : \langle 'SmokeAttribute', Boolean, Any, SmokeDifference, Smoke_f \rangle$

An event is formally defined as a n-dimensional space, denoted event space, where each dimension represents an event feature (e.g., time, social, topic, temperature). To create a dimension, one needs to use the following syntax:

```
CREATE DIMENSION <dimension_id> (
        [o =] <value>,
        [datatype =]  <value>,
);
```

Where `dimension_id` is the unique identifier of the dimension, `o` is the origin point of a dimension, specifying the first value on `DIMENSION`, and `datatype` denotes the attribute datatype shared by all values on `DIMENSION`, `datatype` $\in$ `ADT` (the set of all attribute datatypes). Since each attribute datatype is mapped

to a feature, each dimension is also mapped to/represents a specific event feature. For example, the following describes five event space dimensions, each having an identifier, an origin value, and an attribute data type (and therefore an associated event feature). These event dimensions help define the event space of a fire event:

- $Time$: ⟨1, 30/12/2017 1:30 pm, $Time_{adt}$⟩
- Geo : ⟨2, $Paris$, Geo_{adt}⟩
- $Temp$: ⟨3, 20 (degrees Celsius), $Temp_{adt}$⟩
- $CO2$: ⟨4, 250$PPM(PartsPerMillion)$, $CO2_{adt}$⟩
- $Smoke$: ⟨5, No, $Smoke_{adt}$⟩

And finally, the following query creates an event space:

```
CREATE eSPACE <eventSpace_id> (
        [D =] {<value>},
        [SO =] {<value>},
);
```

Where `eventSpace_id` is the unique identifier of the event space, `D` is a set of dimensions that constitute the space (such as `D` at least contains two dimensions: temporal and spatial), and `SO` is the set of data objects that belong to the event space (the list of objects is empty at the space creation, after the detection process all data objects are inserted into their respective spaces).
Since all events are heavily mapped to a certain time period and a geo-location, we ensure that the event space is at least two dimensional (i.e., at least the temporal and spatial dimensions exist). The additional dimensions that define the event represent the features that the expert chose. This way the framework is re-usable in different contexts (criterion 1 cf. Section 2) and the user is able to customize the event definition in order to get more interesting results (criterion 2 cf. Section 2). In addition, he/she specifies the feature (or set of features) that he/she would like to consider as key for the feature-centric event detection.
For example, $eSpace$: ⟨1, ($Time$, Geo, $Temp$, $CO2$, $Smoke$), SO⟩ defines a fire event where:

- 1 is the id of $eSpace$
- The five dimensions that define the fire event are:
  - $Time$: ⟨1, 30/12/2017 1:30 pm, $Time_{adt}$⟩
  - Geo : ⟨2, $Paris$, Geo_{adt}⟩
  - $Temp$: ⟨3, 10 (degrees Celsius), $Temp_{adt}$⟩
  - $CO2$: ⟨4, 250(PPM : $PartsPerMillion$), $CO2_{adt}$⟩
  - $Smoke$: ⟨5, No, $Smoke_{adt}$⟩
- SO is the set of data objects sensed/shared during the fire event, $forall$ $so \in SO$, so has five coordinates (temporal, spatial, temperature, CO2, and Smoke).

A domain expert can also execute delete and update related queries:

```
DELETE [FEATURE|ADT|DIMENSION|eSPACE] <id> (
WHERE Condition,
);

UPDATE [FEATURE|ADT|DIMENSION|eSPACE] <id> (
SET [field_name =] <new value>,
WHERE Condition,
);
```

Once the event definition is established, the Pre-processor module requests event related data from the storage unit (i.e., from the object repository). The purpose of this step is to analyze the attributes of each data object (Def. 3). An attribute is defined as a value associated with an attribute data type. We define a data type function denoted dt, that returns the attribute data type of a value based on the data object attributes.

Definition 3 *A **Data Object** is defined as a 2-tuple, so* : $\langle id, V \rangle$*, where:*

- *id is the unique identifier of a data object*
- *V is a set of attribute values according to a given ADT, such that* $\forall a_i \in ADT \quad \exists v_i \in V \mid dt(v_i) = a_i$. ■

The *Pre-processor* will extract objects having attributes related to the features found in the event space. For instance, if one targets sports events having temporal, spatial, and a (sport) topic features, the *Pre-processor* extracts from the data object repository (cf. Fig.3) all objects having the following attributes: (i) a temporal; (ii) a geo-location; and (iii) a sports-related tag/annotation. Finally, the selected data objects are sent to the Attribute Extractor module.

Attribute extraction: In this step, the event definition, provided by the incoming event space (*eSpace*), is essential for knowing which data object attributes should be extracted and included in the rest of the processing. The attribute extraction objective is to examine the dimensions that constitute *eSpace* and select the list of attribute data types needed for event detection. The Attribute Extractor module (cf. Fig.4) initiates a cleaning process via the Converter sub-module in order to have the same units for data object attributes (e.g., having all temperature values in Celsius). The cleaned data objects are stored in the data objects repository (cf. Fig.3). Finally, from every data object, the Extractor sub-module extracts the needed attributes (based on the event definition). Both data objects and their attributes will be used in the following steps for lattice construction.

Lattice Construction: In this step, an event agent processes the previously extracted attributes, and data objects into lattice attributes and objects, in order to

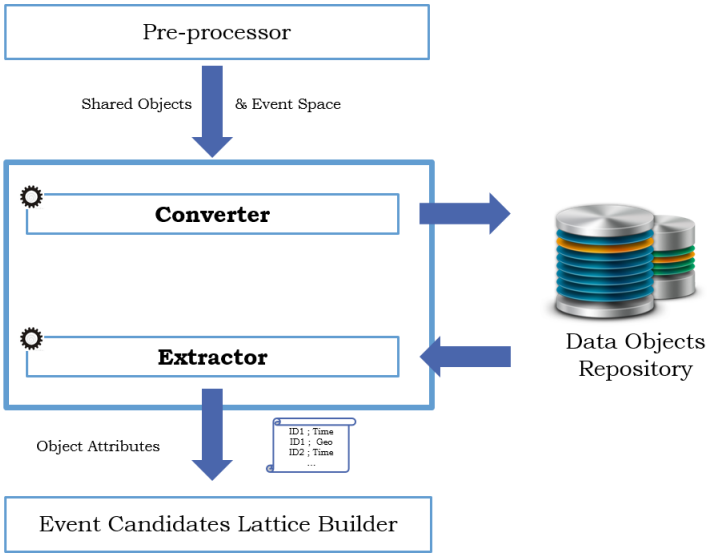

Fig. 4: The Attribute Extractor module

generate one output: the lattice. The *Feature-centric Event Lattice Builder* is the FCA backbone. It integrates the four step process of FCA clustering described in Section 3.2. To do so, we define lattice attribute types in Def.4. These types will be used when defining the lattice attributes (cf. Def.5). Lattice attributes, as defined here, ensure that any given object/attribute can be represented in the FCA formal context. Therefore, any object having attributes can be properly integrated in the clustered data set. This allows the event detection process of eVM to be generic and applicable in various contexts (e.g., social events, sensor events). Finally, for object/lattice attribute mapping, we define a binary cross rule denoted BXR (cf. Def.6). This process is repeated for each event detection run.

Definition 4 ***lat*** *is a lattice attribute type representing an interval* $[a, b[$ *where* $lat : \langle a, b, T \rangle$*, where:*

- *a is the lower boundary value*
- *b is the upper boundary value*
- *T is a value representing the period having a primitive data type of either integer or float, such that:*
  - $dt(a) = dt(b) \in ADT$ *and*
  - $b = a + T$. ■

Definition 5 *A lattice attribute, denoted* ***la****, is defined as a 4-tuple* $la : \langle f, eSpace, lat, y \rangle$ *where:*

- $f \in F$ *is the event feature mapped to lattice attribute la*
- *eSpace is the event space in which the detection will take place*
- *lat (cf. Def. 4) is the lattice attribute type*

– *y is a granularity* | $y \in f.G$ *and*

$$lat.T = \begin{cases} y & \text{if } f.interval = True \\ 0 & Otherwise \end{cases}$$

$lat.a = so_i.v_j$, *where:*

- $so_i \in eSpace.SO$ *and*
- $(v_j \in so_i.V) \wedge (dt(v_j).f = f)$.

■

For example, from the fire event example, we can find the following lattice attributes:

– Time intervals
– Geo locations
– Temperature intervals
– CO2 intervals
– Smoke existence (or not)

Definition 6 *A binary cross rule, denoted as* ***BXR****, is defined as a function that maps a shared object x to its respective lattice attribute y where* $x.v_i \in x.V$*:*

$$BXR = \begin{cases} 1 & \text{if } (y.lat.T = 0 \wedge y.lat.a = x.v_i) \vee \\ & (y.lat.T \neq 0 \wedge x.v_i \in [y.lat.a, y.lat.b[) \\ 0 & Otherwise \end{cases}$$

■

Then the *Feature-centric Event Lattice Builder* constructs the FED (Feature-centric Event Detection) formal context, denoted ***ffc*** (cf. Def. 7). Once the ***ffc*** is created, formal concepts (cf. Fig. 6) are extracted and a lattice (cf. Fig. 7) is generated. This process is described in steps 2-4 of Section 3.2. This lattice is called an Event Candidate Lattice, where each node is a potential feature-centric event. Fig. 5 illustrates the inner composition of the Event Candidates Lattice Builder module.

Definition 7 *A FED Formal Context, denoted* ***ffc****, is defined as a 6-tuple* ffc : $\langle eSpace, F, f_{LAG}, X, Y, I \rangle$*, where*

– *eSpace is the event space in which the detection takes place*
– *F is the set of one event features*
– f_{LAG} *is the function that generates the lattice attributes, described in Algorithm 1*
– $X = eSpace.SO$ *is the set of shared objects*
– $Y = \bigcup_{i=0}^{|X.V|-1} \{la_i\}$ *is the set of lattice attributes* | $X.V = \bigcup_{\forall so \in X} \{so.V\}$ *is the union of all attribute values from the shared objects in eSpace*
– *I is a BXR(x,y) where* $x \in X \wedge y \in Y$. ■

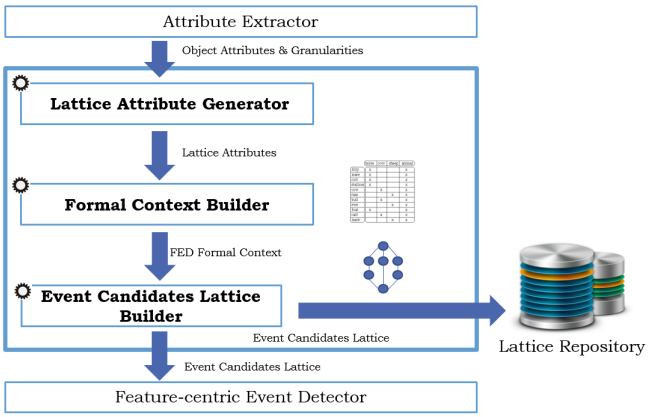

Fig. 5: The Event Candidates Lattice Builder module

To follow up with the fire event example, Table 3 illustrates how lattice attributes (columns) are mapped to incoming sensor observations (rows) using the binary cross rule in the FED formal context.

Table 3: Fire Event Formal Context Example

	Time		Geo		Temp		CO2		Smoke	
	[9 - 9:15[	[9:15 - 9:30[	Loc1	Loc2	[20 - 40[	[40 - 60[	[250 - 350[	[350 - 450[	Yes	No
O1	X		X			X		X	X	
O2	X		X			X		X	X	
O3	X		X			X		X	X	
O4		X		X	X		X			X
O5		X		X	X		X			X
O6		X		X		X	X			X

The example in Table 3 shows six sensor observations (objects), mapped to their respective attributes. For instance, observation 1 has a timestamp value between 9 and 9:15 AM, therefore it is mapped to the lattice attribute $[9-9:15[$. This observation is taken from a sensor deployed in Loc1 and has a temperature reading that is included in the $[40-60[$ degrees Celsius interval. Moreover, the other five observations are also mapped to their corresponding attributes using the binary cross rule. This represents the (FED) formal context in this scenario. In Algorithm 1, we detail the lattice attribute generation process. This starts by extracting all object attribute values (lines 5-11). If the value is mapped to a feature that is generated as an interval (e.g., *time*), the algorithm calls the Create-Intervals function (lines 19-23). If not (e.g., *social*), the algorithm generates a lattice attribute type having a null period and creates the corresponding lattice attribute (lines 13-18). This step allows the creation of generic lattice attributes from various features, thus providing extensibility (criterion 2). Algorithm 2 details the Create-Intervals function. This process extracts all values related to the same feature (lines 4-9), orders them (line 10), selects a minimum and a maximum value (lines 11-12), and creates periodic intervals starting from the minimum to the maximum value (lines 14-22). The period is calculated based

on the chosen feature granularity (line 15). This makes the detection more user-centric (criterion 1). Finally, the result is added the the output of Algorithm 1.

Algorithm 1: Lattice Attribute Generation (cf. Def. 7 - f_{LAG})

```
1  Input: eSpace
2  Output: RES                                          // List of all lattice attributes
3  VAL = new List()                                     // Shared Objects attribute values list
4  PD = new List()                                      // Processed event features list
5  foreach so ∈ eSpace.SO do
6  |  foreach v ∈ so.V                                 // This loop extracts all object attribute values
7  |  do                                                from all objects in eSpace and stores them in
8  |  |  if (v ∉ VAL) then                             the VAL list
9  |  |  |  VAL←v
10 |  end
11 end
12 foreach v ∈ VAL do
13 |  if (not dt(v).f.Interval)                        // If the value is not generated as an
14 |  then                                              interval
15 |  |  lat ← LAT(v, lat.a + lat.T, 0)
16 |  |  la ← LA(dt(v).f, eSpace, lat, dt(v).f.g)      // Create la with lat.T=0
17 |  |
18 |  |  RES ← la
19 |  else
20 |  |  if (dt(v).f ∉ PD) then
21 |  |  |  RES ← (Create-Intervals(VAL,v,PD, eSpace))  // Call
22 |  |  |                                               Create-Intervals
23 |  end                                               function
24 end
25 return RES
```

Algorithm 2: Create-Intervals

```
1  Input: VAL, v, PD, eSpace                            // Input provided by Algorithm 1, line 21
2  Output: LAI                                          // Generated lattice attributes intervals
3  int i = 0
4  TEMP = new List()                                    // Temporary object attribute list
5  foreach val ∈ VAL do
6  |  if (dt(val).f == dt(v).f)                         // Extract all object attribute values
7  |  then                                              having the same feature as v and store
8  |  |  TEMP ← val                                    them in TEMP
9  end
10 Order_ascending(TEMP)                                // Order TEMP ascending
11 min ← TEMP.get(0)                                    // min is the first element of TEMP
12 max ← TEMP.get(|TEMP| − 1)                           // max is the last element of TEMP
13 lat ← LAT()
14 while (lat.b < max) do
15 |  lat ← LAT(min, lat.a + (i+1) × lat.T, dt(v).f.g)
16 |  if (lat.b > max)                                  // This loop creates
17 |  then                                              intervals of period
18 |  |  lat.b ← max                                   lat.T = f.g
19 |  la ← LA(dt(v).f, eSpace, lat, dt(v).f.g)         (feature
20 |  LAI ← la                                         granularity)
21 |  i++
22 end
23 PD ← dt(v).f                                         // Add feature to the list of processed features
24 return LAI
```

Fig. 6, shows the formal concepts extracted from the fire event FED formal context using the FCA operators. And finally, the generated lattice is shown in Fig. 7 where nodes are potential fire events.

Formal Concepts of "Fire Event Example"	
1	<{O1,O2,O3}, {[9:00 - 9:15[,Loc1,[40 - 60[,[350 - 450[,Yes}>
2	<{O4,O5}, {[9:15 - 9:30[,Loc2,[20 - 40[,[250 - 350[,No}>
3	<{O6}, {[9:15 - 9:30[,Loc2,[40 - 60[,[250 - 350[,No}>
4	<{O1,O2,O3,O6}, {[40 - 60[}>
5	<{O4,O5,O6}, {[9:15 - 9:30[,Loc2,[250 - 350[,No}>
6	<{O1,O2,O3,O4,O5,O6}, {}>
7	<{}, {[9:00 - 9:15[,[9:15 - 9:30[,Loc1,Loc2,[20 - 40[,[40 - 60[,[250 - 350[,[350 - 450[,Yes,No}>

Fig. 6: Fire Event Formal Concepts

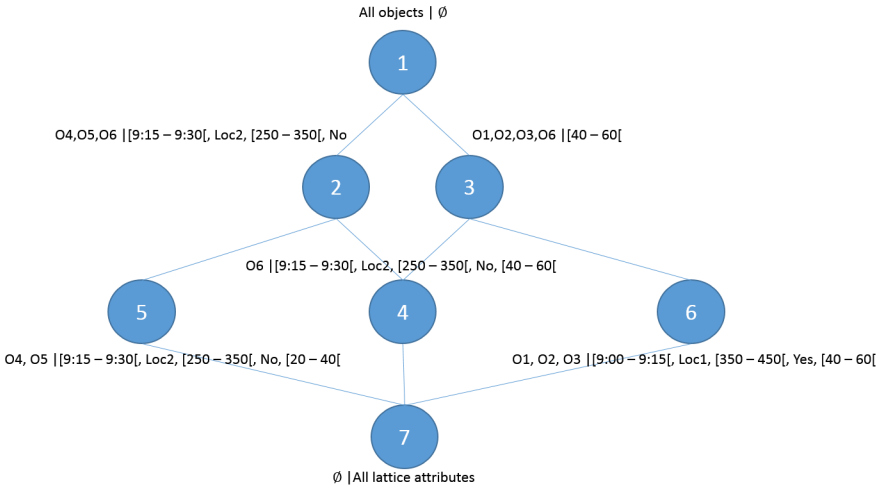

Fig. 7: Fire Event Lattice

Event Detection: The *Feature-centric Event Detector* module uses the previously generated lattice, an event detection rule (an operator that checks the lattice nodes in order to find the targeted events), and the features in order to detect feature-centric events (cf. Def. 8). We define a default event detection rule, as a set of lattice attributes that comply with the two conditions mentioned in Def. 8. The rule is extensible, thus allowing the integration of multiple event features (e.g., *Time*, *Geo-location*, *Social*, *Topic*), each represented by the corresponding lattice attribute. This rule (cf. Fig.8.(a)) uses the selected key features in order to target the related feature-centric events. For example, the rules illustrated in Fig.8.(b), 8.(c), 8.(d), and 8.(e) detect user, geo, topic, and time-centric events respectively (in a social event detection context). For instance, Fig. 8.b detects social events (anytime, anywhere, and any topic) of each specific person. Fig. 8.c is another example of a detection rule where the key feature is the geo location, i.e., this rule detects events that happened anytime, with any person, and concerning any topic at a specific location. These rules reflect the targeted events'definitions and are extensible. Features can be added/removed

(e.g., temperature, CO_2 levels, smoke) for specific event detection needs (e.g., fire events could be defined as events having high temperature, CO2 and smoke at any time and place. Therefore, node 6 in Fig. 7 is detected as a fire event). Finally, for testing purposes, developers can change/add detection rules using the *Rule Selector* module. Since the lattice is not affected by the rule change, only the event detection step is repeated based on the new detection rule.

Feature	Condition
f1	Value Range
f2	Value Range
f3 (key)	Specific value
...	...

(a)

Feature	Condition
Time	Time Range
Geo	Geo Range
Social (key)	Specific value
Topic	Topic Range

(b)

Feature	Condition
Time	Time Range
Geo (key)	Specific Value
Social	Social Range
Topic	Topic Range

(c)

Feature	Condition
Time	Time Range
Geo	Geo Range
Social	Social Range
Topic (key)	Specific Value

(d)

Feature	Condition
Time (key)	Specific Value
Geo	Geo Range
Social	Social Range
Topic	Topic Range

(e)

Fig. 8: Default detection rule

Definition 8 *A feature-centric Event, denoted* ***fce****, is a Formal Concept defined as a 4-tuple* $fce : \langle ffc, central_F, A, B \rangle$*, where:*

- *ffc is a FED Formal Context (Def. 7)*
- $central_F$ *is the set of selected key features* $|central_F \subseteq ffc.F$
- A *is a set of data objects* $\mid A \subseteq ffc.X$
- B *is a set of lattice attributes* $\mid B \subseteq ffc.Y$ *where* $\forall b_i, b_j \in B \wedge i \neq j$*:*
  - ***Condition 1:*** $b_i.f \neq b_j.f$
  - ***Condition 2:*** *if* $b_i.f.label = c_f.label | \forall c_f \in central_F$*, then* $dist(b_i.lat.a, so_j.v_k)=0 \mid \forall so_j \in A \wedge \forall v_k \in so_j.V, \quad dt(b_i.lat.a) = dt(so_j.v_k)$.■

Finally, Fig. 9 details the interaction between the Rule Selector module and the Feature-centric Event Detector module. A detection rule change can be requested through the Event Query Language. One can create a new rule, select, or update an existing one. Based on the developer's choice, the Rule Validator sub-module checks the syntax of the newly created or updated rule prior to storage. If one decides to select an existing rule, the Rule Selector sub-module returns the chosen rule. In both cases, the event detection is repeated using the chosen rule and new results are generated.

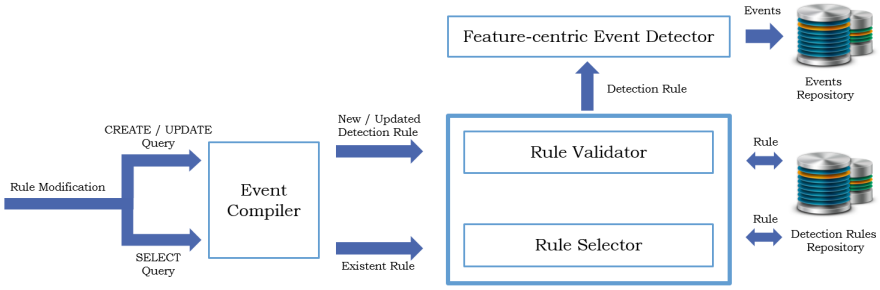

Fig. 9: The Rule Selector module

The syntax to create, select, or update a detection rule (DR) is described below:

```
CREATE DR <dr_id> (
[f_id =] {<feature_id>},
[key_id =] {<feature_id>},
);

SELECT DR <dr_id> FROM Detection_Rules_Repository(
WHERE Condition,
);

UPDATE DR <dr_id> (
SET [f_id =] {<feature_id>},
WHERE Condition,
);
```

Where f_id is a list of features (identified by their respective ids) that are considered by the detection rule, and key_id is the set of key features considered by the detection rule.
Finally, event spaces (eSPACE) do not initially contain any data objects (SO) because they are created prior to the event detection process. Once the latter occurs, an insert query adds each data object to its corresponding event space. The following describes the syntax of an insert query:

```
INSERT INTO [eSPACE] <id> (
{<value>},
WHERE Condition,
);
```

4 Implementation & Evaluation

We wanted to evaluate how generic and re-usable the framework is. Therefore, we instantiated from the eVM two applications: the first, for social event detection, and the second for conflict event detection. These two contexts provide a variety of features each (social features, conflict related features), and different data objects (multimedia shared data in the social context, news stories in the conflict events context). More particularly, we aimed to validate the components related to the Event Detection part of the eVM framework (Event Detector modules cf. Fig.3). In order to do so, we measured, for each application (social/-conflict event detection), the event detection accuracy (with the integration and adaptation of FCA). We also evaluated the algorithm's performance based on

execution time and memory consumption. The objectives of the experimentation were the following: (i) show that the approach is re-usable (criterion 1) by detecting feature-centric events in different contexts (e.g., social event detection, conflict event detection); (ii) measure the impact of domain specific expertise (criterion 2) on the detection accuracy (regarding the choice of event features and granularities in the event definition phase) while reducing human intervention (criterion 9); (iii) demonstrate that eVM is multi-modal (criterion 6) and multi-source (criterion 7) by measuring the impact of adding various features on the performance from various data sources; and (iv) proving that eVM is accurate when given (by the domain experts) optimal features/ granularities, and event definition. We do not aim at comparing accuracy results with other works since the objective is to provide a re-usable, easy to integrate, accurate backbone for event detection, that can be adjusted/configured by domain experts (i.e., by defining events, detection rules, features, granularities). We do not present here the evaluation results related to (i) the incremental processing (even though FCA is incremental [39]); (ii) evolution; (iii) extensibility; and (iv) ease of integration. This will be presented in a separate work.
Regarding the implementation, we developed a platform that integrates a front end mobile application and a cloud-based back end. In order to have a mobile application for both Android and iOS users, we used Visual Studio and Xamarin[3], a mobile development platform that allows building native mobile applications from a shared C# code. Xamarin also provides the following features: (i) complete binding for the underlying SDKs for both Android and iOS; (ii) Objective-C, Java, C, and C++ interoperability; (iii) mobile cross platform support; (iv) advanced base class library; and (v) Xamarin.Forms maximizes code sharing for cross-platform development, Xamarin.iOS and Xamarin.Android provide direct access to platform-specific APIs (cf. Fig.10-11).

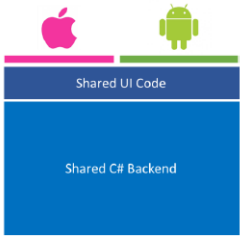

Fig. 10: Cross Platform Coding

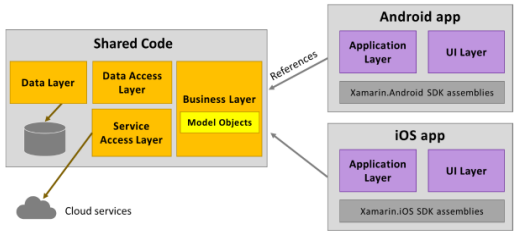

Fig. 11: Xamarin Forms Architecture

We detail next the implementation of the two instances of the framework. Then, we show the evaluation of performance and accuracy by detecting feature-centric events in different contexts:(i) Social (e.g., wedding, birthday) Event Detection; and (ii) Conflict (e.g., wars, protests) Event Detection. In each context, we detail the corresponding dataset, and performance and accuracy tests/results.

[3] https://developer.xamarin.com

4.1 Instantiating applications from the eVM framework

As previously mentioned, we created two instances from the eVM framework. The first is dedicated to feature-centric social event detection. In this case, the targeted events were defined based on various social-related features such as participants and topics in addition to time and location. The data objects, i.e., the event related data that need to be clustered into feature-centric events, were considered to be photos and videos taken/shared during the social events by participants. The second instance is designed for feature-centric conflict event detection. In this case, the targeted events were defined based on time, geo-location, and a set of contextual features such as the actors (i.e., aggressor, defender), the news source that covered the conflict, the conflict type (e.g., protest, war, planned attack), and finally the number of casualties. The data objects are considered news stories regarding the targeted conflict events.

4.2 Algorithm Evaluation

To evaluate the re-usability of our approach, we detected feature-centric events in different contexts. First, we detail our experimentations and results for the social event detection application (Test 1), then for the conflict event detection application (Test 2). The performance tests were conducted on a machine equipped with an Intel i7 2.60 GHZ processor and 16 GB of RAM. The aim was to test the performance of our eVM Event Detector algorithm.

Test 1: Application to Social Event Detection
ReSEED Dataset: To evaluate the detection results, we used the ReSEED Dataset, generated during the Social Event Detection of MediaEval 2013 [34]. It contains real photos crawled from Flickr, that were captured during real social events which are heterogeneous in size (cf. Fig. 12) and in topics (e.g., birthdays, weddings). The dataset contains 437370 photos assigned to 21169 events. In our evaluation, we used three event features: *time, location, and social*, since ReSEED photos have *time, geo, and social* attributes. In ReSEED, 98.3% of photos contain *capture time* information, while only 45.9% of the photos have a *location*. We had to select photos having these attributes from the dataset. This left us with 60434 photos from the entire dataset. In ReSEED, the ground truth used for result verification assigns photos to social events. Since, our approach is focused on feature-centric events (in this experimentation, user-centric events), we modified the ground truth to split the social events into their corresponding user-centric events. Since the splitting is based on the event features, we need to specify the feature granularities during the process. The latter are not specified in ReSEED, therefore we chose the lowest granularity values: *day* for *time*, *street* for *geo*, and *photo creator name* for *social*. The ground truth refactoring process is described in Fig. 13. First, we extracted the photos of each event in the ground truth. Second, we used the *timestamps of photo capture* to group photos by *day*. Third, we split the resulting clusters into distinct groups based on *street* values.

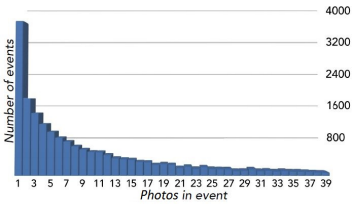

Fig. 12: ReSEED Photo distribution

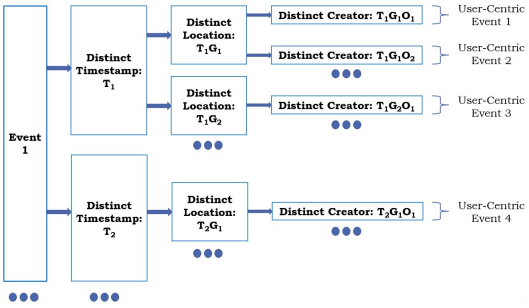

Fig. 13: Refactoring ground truth

Finally, the result was further split based on distinct *photo creators*.

Performance Evaluation: We considered two criteria for this task: (i) total execution time and (ii) memory overhead.

Use Cases: The performance is highly affected by the number of photos, generated attributes, and clusters. We noticed that granularities *day* for *time* and *street* for *geo* generate more clusters and attributes than any other granularity combination. Therefore, we used *day* and *street* to test the prototype's performance in three worst case scenarios:

- Case 1: We selected the biggest event (1400 photos) as input. We varied the number of photos progressively from 1 to 1400. Since all photos are related to one event, the number of detected clusters should be one.
- Case 2: We extracted 400 events each having exactly one photo. We varied the number of photos from 100, 200, 300 to 400. The number of generated clusters for each iteration should be 100, 200, 300, and 400 respectively.
- Case 3: The goal is to test with as many photos as possible related to different events. We varied the number of photos from 15000, 30000, 45000 to 60434. Since thousands of events contain only one or two photos per event (worst case scenario), this case will generate the most clusters.

Results & Discussion: In Cases 1 and 2 (Figures 14.a and 14.b), where the number of photos does not exceed 1400 and 400 respectively, the total execution time is quasi-linear. However, in Case 3 (Figure 14.c), we clustered the entire dataset (60434 photos). The total execution time tends to be exponential, in accordance with the time complexity of FCA. When considering RAM usage, we noticed a linear evolution for the three cases (Figures 14.d, 14.e, and 14.f). RAM consumption is significantly higher in Case 2, where we generated 400 clusters, than in Case 1, where we generated one cluster. In Case 3, RAM consumption is the highest because both the number of photos at the input, and the number of generated clusters (detected events) were the highest. Other tests were conducted, Fig.15 (left) shows that low granularities (e.g., day) consume more execution time than high ones (e.g., year). This is due to the generation of more lattice attributes and clusters. In addition, Fig.15 (right), shows that considering

more features in the processing is also more time consuming. Nonetheless, the evolution from one to three features remains quasi-linear, making the process extensible.

Accuracy Evaluation: We chose to consider the criteria proposed by Medi-

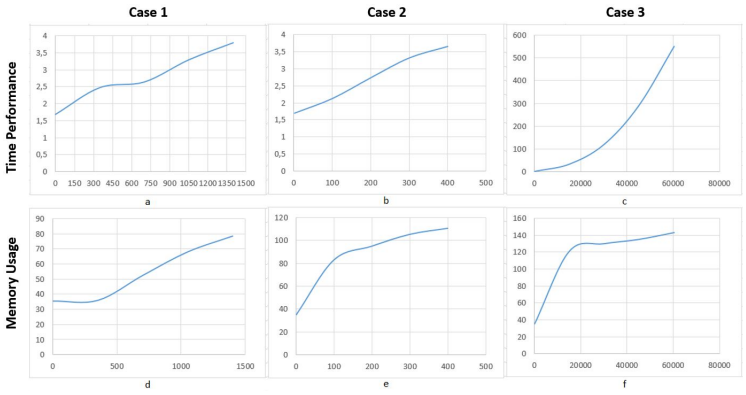

Fig. 14: Performance Results

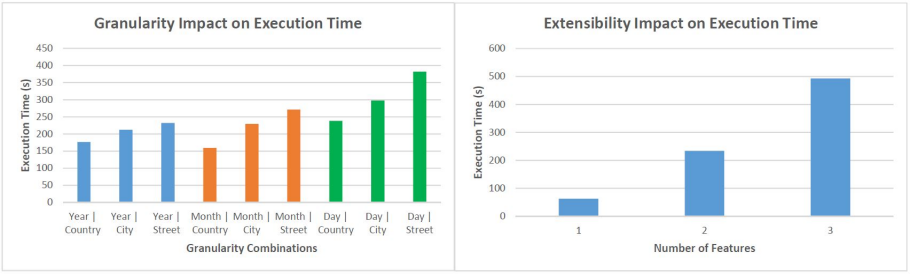

Fig. 15: Granularity and Extensibility Impact

aEval for clustering quality evaluation. We calculated the F-score, based on the Recall (R) and Precision (PR), and the Normalized Mutual Information (NMI) using ReSEED's evaluation tool. These criteria are commonly adopted in information retrieval and social event detection. A high F-score indicates a high quality of photo to user-centric event assignment while NMI will be used to measure the information overlap between our clustering result and the ground truth data. Therefore, a high NMI indicates accurate clustering result.

Use Cases: Since we considered the *time*, *geo*, and *social* features, we identified all possible combinations of the detection rule (see Table 4). In order to test granularity impacts, Table 5 sums up the different granularity combinations.

When applying detection rules to granularity combinations, we get 63 use cases. We measured for each one the NMI and F-Score.

Results & Discussion: Results shown in Table 6, highlight the following:

Table 4: Detection Rule

Combination	Number of Features	Features Considered in the Detection Rule
1	3	Time, Geo, Social
2	2	Time, Geo
3		Time, Social
4		Geo, Social
5	1	Time
6	1	Geo
7	1	Social

Table 5: Granularity Combinations

Combination	Granularities: Time / Geo
1	Year / Country
2	Year / City
3	Year / Street
4	Month / Country
5	Month / City
6	Month / Street
7	Day / Country
8	Day / City
9	Day / Street

(i) *Detection rule/features impact:* The detection rule based on *time*, *geo*, and *social* features generates the highest NMI and F-score (NMI: 0.9999 and F-Score: 0.9995). It also exceeds all other detection rules (e.g., the one including solely *time* and *geo* features) in every granularity combination. This underlines that eVM can cope with various features such as the *social* feature in the detection task. Moreover, it highlights eVM's multi-modality, which allows the integration of additional features (having different datatypes) and the accurate detection of user-centric events.

(ii) *Granularity impact:* The results improve, when the clustering is based on granularities closer to the ones used in the ground truth. For example, in the case of granularities *year, country*, the F-Score achieved based on *time* and *geo* features is 0.1911, but for the detection rule that considers only the *social* feature the F-Score is higher: 0.5376. This is because the granularities for *time* and *geo* are the most general (*year and country*). Therefore, the impact factor of granularities is more important than that of the number of features considered in the detection rule. Some rules can exceed others for specific granularity combinations (e.g., *Time Geo* exceeds *Time Social* and *Geo Social* for granularities Year/Month/Day-Street while Time Social exceeds the other two rules for Year/Month/Day-Country). The best result can be achieved by considering the maximal number of features having correct granularities. This indicates that the granularities should not be fixed for all scenarios. When given the best granularities, our approach detects the user-centric events very accurately.

Test 2: Application to Conflict Event Detection

ACLED Dataset: The aforementioned experiments targeted feature-centric social events which have features such as time, location, social (e.g., participants), and topics (e.g., birthday, marriage). In the following experiments, we targeted feature-centric conflict events. The latter, have different features than social events (e.g., time, location, aggressor, defender, press news source, conflict type, casualties). This allowed to have a different event definition, as well as test the

Table 6: Clustering Results

Detection Rule	**Measure**	**Granularities**								
		Year			Month			Day		
		Country	City	Street	Country	City	Street	Country	City	Street
Time Geo Social	F-Score	0.6399	0.8180	0.8662	0.7964	0.8619	0.8948	0.9535	0.9742	**0.9995**
	NMI	0.9181	0.9602	0.9729	0.9549	0.9703	0.9789	0.9880	0.9938	**0.9999**
Time Geo	F-Score	0.1911	0.7678	0.8473	0.4943	0.8367	0.8821	0.8854	0.9542	0.9892
	NMI	0.7113	0.9475	0.9684	0.8707	0.9637	0.9759	0.9729	0.9894	0.9977
Time Social	F-Score	0.6245	0.6245	0.6245	0.7939	0.7939	0.7939	0.9534	0.9534	0.9534
	NMI	0.9143	0.9143	0.9143	0.9544	0.9544	0.9544	0.9879	0.9879	0.9879
Geo Social	F-Score	0.5085	0.7718	0.8357	0.5085	0.7718	0.8357	0.5085	0.7718	0.8357
	NMI	0.8742	0.9470	0.9653	0.8742	0.9470	0.9653	0.8742	0.9470	0.9653
Time	F-Score	0.0220	0.0220	0.0220	0.1399	0.1399	0.1399	0.7278	0.7278	0.7278
	NMI	0.3971	0.3971	0.3971	0.7069	0.7069	0.7069	0.9392	0.9392	0.9392
Geo	F-Score	0.0559	0.6958	0.8343	0.0559	0.6958	0.8343	0.0559	0.6958	0.8343
	NMI	0.5084	0.9241	0.9646	0.5084	0.9241	0.9646	0.5084	0.9241	0.9646
Social	F-Score	0.5376	0.5376	0.5376	0.5376	0.5376	0.5376	0.5376	0.5376	0.5376
	NMI	0.8755	0.8755	0.8755	0.8755	0.8755	0.8755	0.8755	0.8755	0.8755

accuracy of the detection process in different contexts. For this purpose, we used the ACLED[4] (Armed Conflict Location & Event Data Project) Dataset. It is a disaggregated conflict collection, analysis and crisis mapping project. ACLED collects the dates, actors, types of violence, locations, and fatalities of all reported political violence and protest events across Africa, South Asia, South East Asia and the Middle East. Political violence and protest include events that occur within civil wars and periods of instability, public protest and regime breakdown. The dataset contains event records that span over years. We selected 49000 events, from the African subset, that date from April 1998 till January 2018. For each event, we generated shared objects having twelve attributes each (a object owner (press news source), a latitude, a longitude, a country, a region, a city, a street, a datetime, a conflict type, two actors, and the number of casualties). In total, we tested the accuracy of the detection process, the impact of the number of included features on the performance based on an input of 50000 shared objects related to the events mentioned above.

Performance Evaluation: To give a detailed view of the algorithm's performance, we measured the impact of: (i) the number of objects at the input; and (ii) the number of included features on the execution time.

Use Cases: The performance is affected by the input size, i.e., the number of objects to be processed, and the number of event features included in the clustering. Therefore, we experimented the following cases:

– Case 1: We ran the detection module five times and measured the execution time of each run. Every iteration has the following configuration: (i) all seven features are considered in the clustering; and (ii) the granularity choices for time and geo-location are day and street respectively. The only variable is the input size. The first run processes 10000 objects, the second 20000, the third run 30000, then we considered 40000 in the fourth run, and finally 50000 in the last run.

[4] https://www.acleddata.com

– Case 2: We ran the detection module seven times and measured the execution time of each run. Every iteration has the following configuration: (i) the input size is the same, 50000 objects (the entire dataset); (ii) the granularity choices for time and geo-location are day and street respectively. In the first run, we only consider one feature (time) in the clustering. Then, for each iteration, we include one additional feature (e.g., time and geo-location in the second run). The last run includes all seven features (time, geo-location, press news source, actor 1, actor 2, conflict type, and casualties).

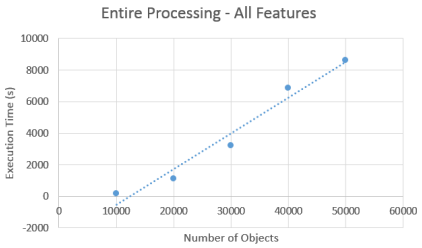

Fig. 16: Case 1 Results

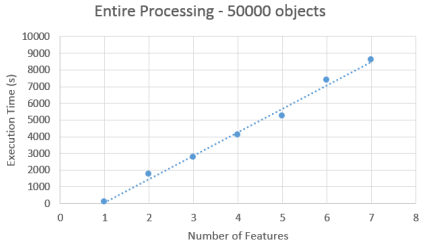

Fig. 17: Case 2 Results

Results & Discussion: Fig.16 shows the impact of augmenting the size of the input on the algorithm's execution time (Case 1). We notice that the evolution of execution time is quasi-linear. Fig.17 shows the impact of including more features in the processing on the execution time (Case 2). The evolution is also quasi-linear in a worst case scenario from a granularity and input size point of view. We also analyzed the execution time of each step of the event detector (i.e., attribute extraction, lattice construction, event detection cf. Section 3). The highest cost in terms of execution time is related to the event detection step (not FCA computation), which consists of scrolling through the nodes of the lattice in order to select nodes that comply with the chosen feature-centric event definition. Results can be optimized by looking into better ways of scrolling through the lattice (graph analysis techniques). This will be conducted in a future work.

Table 7: Detection Rule Combinations

Combination	Number of features	Considered features
1	3	Time, Geo, Press News Source
2	2	Time, Geo
3		Time, Press News Source
4		Geo, Press News Source
5	1	Time
6	1	Geo
7	1	Press News Source

Accuracy Evaluation: We used the same metrics (F-Score & NMI) for accuracy evaluation. We tested the same granularity combinations shown in Table 5 for time and geo-location features. We did not vary the granularities of the five other features (press news source, actor 1, actor 2, conflict type, and casualties) found in the ACLED dataset. As for the detection rule, we limited the combinations to three features: (i) time; (ii) geo-location; and (iii) press news source (chosen as central feature in this experimentation). This could be extended to include the actors of events (e.g., to detect the group most involved in armed conflicts), the number of casualties (e.g., to detect the deadliest conflict events), and the conflict types (e.g., to compare occurrence of protests between countries). The detection rule combinations are detailed in Table 7. Finally, we used the same tool (provided by the ReSEED work) for F-Score and NMI calculation. Accuracy results are detailed in Table 8.

Table 8: ACLED Accuracy Results

Detection Rule	Measure	Granularities											
		Year				Month				Day			
		Country	Region	City	Street	Country	Region	City	Street	Country	Region	City	Street
1	F-Score	0.3194	0.6181	0.6737	0.732	0.5012	0.7598	0.7873	0.8195	0.8439	0.9454	0.9528	**0.9631**
	NMI	0.8234	0.9287	0.9392	0.9515	0.9011	0.9604	0.9645	0.9701	0.9764	0.9923	0.9933	**0.9949**
2	F-Score	0.0254	0.3366	0.4385	0.5465	0.1789	0.6257	0.6771	0.7374	0.7252	0.921	0.9325	0.9477
	NMI	0.6652	0.8668	0.8885	0.9144	0.8263	0.9382	0.9461	0.9567	0.9598	0.9889	0.9904	0.9928
3	F-Score	0.2564	0.2564	0.2564	0.2564	0.429	0.429	0.429	0.429	0.8269	0.8269	0.8269	0.8269
	NMI	0.7892	0.7892	0.7892	0.7892	0.886	0.886	0.886	0.886	0.9742	0.9742	0.9742	0.9742
4	F-Score	0.232	0.5028	0.5751	0.6522	0.232	0.5028	0.5751	0.6522	0.232	0.5028	0.5751	0.6522
	NMI	0.7509	0.897	0.9133	0.9327	0.7509	0.897	0.9133	0.9327	0.7509	0.897	0.9133	0.9327
5	F-Score	0.001	0.001	0.001	0.001	0.0116	0.0116	0.0116	0.0116	0.291	0.291	0.291	0.291
	NMI	0.4261	0.4261	0.4261	0.4261	0.6671	0.6671	0.6671	0.6671	0.8941	0.8941	0.8941	0.8941
6	F-Score	0.0015	0.0865	0.1809	0.311	0.0015	0.0865	0.1809	0.311	0.0015	0.0865	0.1809	0.311
	NMI	0.4209	0.7434	0.7858	0.836	0.4209	0.7434	0.7858	0.836	0.4209	0.7434	0.7858	0.836
7	F-Score	0.1938	0.1938	0.1938	0.1938	0.1938	0.1938	0.1938	0.1938	0.1938	0.1938	0.1938	0.1938
	NMI	0.6913	0.6913	0.6913	0.6913	0.6913	0.6913	0.6913	0.6913	0.6913	0.6913	0.6913	0.6913

Results & Discussion: Table 8 highlights the following:
(i) *Detection rule/features impact:* The detection rule 1 (based on *time*, *geo-location*, and *press news source* features) generates the highest NMI and F-score (NMI: 0.9949 and F-Score: 0.9631). It also exceeds all other detection rules (e.g., the one including solely *time* and *geo-location* features) in every granularity combination. This underlines that eVM can provide the appropriate way to conduct the clustering and integrate various datatypes related to a multitude of features due to its multi-modality. However, accuracy can still be improved, few objects were assigned to the wrong clusters. These errors can be minimized by including more features in the detection rule (i.e., the actors, conflict type, and casualties).
(ii) *Granularity impact:* We notice here (as we also did for the ReSEED dataset) that when the clustering is closer in terms of granularities to the ground truth, accuracy improves. For example, in the case of granularities *year, country*, the F-Score achieved based on *time* and *geo-location* features is 0.0254, but for the detection rule that considers only the *press news source* feature the F-Score is higher: 0.1938. This is because the granularities for *time* and *geo* are the most general (*year and country*). Some rules can exceed others for specific granularity combinations. The best result can be achieved by considering the maximal number of features having correct granularities. Finally, these results prove that eVM is re-usable since event detection accuracy was high in different event detection contexts.

5 Clustering Techniques

5.1 Clustering Techniques

Many works in different areas (e.g., information retrieval, event detection, image searching and annotation), have evolved around clustering techniques since their introduction in 1975 when John Henry Holland wrote Adaptation in Natural and Artificial Systems, a ground breaking book on genetic algorithms [18]. Unsupervised clustering is considered since in most cases, we detect events from raw data without prior knowledge on the occurring events. Clustering techniques are commonly grouped into four categories [38]:

Prototype-Based Clustering A cluster is a set of objects that are closest (most similar) to the prototype that defines the cluster than to the prototype of any other cluster. A prototype can be the centroid or the medoid depending on the nature of the data (continuous attributes or categorical attributes). For continuous data, a centroid represents the object with the average (mean) values of all objects (points) in the cluster. As for categorical attributes, since a centroid is not meaningful, the prototype is often a medoid, the most representative point of the cluster. For many types of data, the prototype can be regarded as the most central point. Therefore, prototype-based clustering is commonly referred to as center-based clustering. For example, K-means [19] is a prototype-based clustering technique that groups objects based on a specified similarity measure (e.g., Euclidean distance, Manhattan distance, cosine similarity, Jaccard measure) and creates a set of K clusters represented each by a centroid. K-medoids [21] is another example of this clustering category. Instead of calculating means, actual points from the data are picked as representatives (prototypes) of the clusters. Points are associated to the clusters where they are most similar to the prototype. An iterative swapping process between prototypes and non prototype points is done as long as the quality of the clustering is improved.
These methods have low complexities for both time and space. But the algorithms attempt to find a predefined number of clusters (K): the final number of clusters should be known prior to clustering. In addition, for K-means, in order to start the clustering, the user has to choose initial cluster centers (centroids). This is a key step, if these centroids are chosen randomly clustering results can be poor.

Density-Based Clustering A cluster is represented as a dense region surrounded by a low density region. Objects in the low density zones are considered noise while others in high density regions belong to the group limited by the region. For example, DB-Scan[33] produces a partitional clustering based on density measures. This method studies the neighborhood of each point, and partitions data into dense regions separated by not-so-dense regions. To do so, density at a point p is estimated by counting the points within a circle of center p and radius ϵ. Therefore, a dense region is a circle of radius ϵ containing a

minimal number of points.
On one hand, DB-Scan determines automatically the number of clusters, is relatively resistant to noise, and can handle clusters of arbitrary sizes and shapes. On the other hand, since clustering is affected by the specified radius, DB-Scan loses accuracy when the clusters have widely varying densities. Also, with high-dimensional data, defining the densities becomes more difficult and more expensive (in term of computation time and space). Finally, points in the low-density areas are considered noise which means that not all input data will be present in the clusters.

Graph-Based Clustering Data is organized in graphs/hierarchies where nodes are objects and connections among objects are represented as links connecting the nodes. Therefore, a cluster is defined as a connected component, a group of objects that are connected to one another but have no connections to objects from outside the group. For example, Agglomerative Hierarchical clustering is a graph-based clustering method [5]. First, each point is considered as a singleton cluster. Then repeatedly, the closest two clusters (based on similarity/dissimilarity matrices) are merged until a single all-encompassing cluster remains. Hierarchical clustering can also be divisive, this method is symmetrical to the agglomerative technique. In the divisive algorithm, all points are initially assigned to a single cluster and then based on similarity/dissimilarity measures the splitting into different clusters begins, until each point is assigned to a distinct cluster.
The added value of this method is that clusters are nested in a dendrogram (hierarchical structure) which offers a first level of semantic reasoning by exploiting the hierarchy and the inter-cluster relations. In contrast, the method has a high complexity in both time and space. All cluster merges are final, for high dimensional data such as photos, this is considered as a limitation. Since high dimensional data is more complicated, error correction if data is wrongly assigned to a certain cluster is a major issue.

Conceptual Clustering (Shared-property) A cluster is a set of objects that share some properties. For successful clustering, an algorithm would require a very specific definition of a cluster. This means that prior to the clustering, the shared properties that identify a cluster should be defined in order to generate a concept describing a cluster. The process of generating such clusters is called conceptual clustering. Formal Concept Analysis (FCA) is a conceptual, hierarchical clustering method[7][30]. It analyses data based on object/attribute relationships, extracts concepts, and finally orders them in a lattice. The advantage of having a lattice of formally defined Concepts is that it assures a more advanced level of semantic reasoning. In addition, FCA automatically generates a brief description for each cluster. Nonetheless, time and space complexities could cause concerns in some worst case scenarios where every data object forms a formal concept. In this case, exponential complexities become major technical difficulties.

Discussion: Table 9 shows a comparative summary of clustering techniques with respect to our defined, clustering-related, criteria (cf. Section 2). Prototype-based methods require excessive human intervention and the number of clusters prior to the processing. This is a major limitation in an event detection scenario where the total number of events is unknown prior to detection. In addition, these approaches are not multi-modal, multi-source nor incremental. Density-based methods detect automatically the number of final clusters, thus reducing human intervention but they are not multi-modal nor multi-source. These methods do not consider different types of data at once. In addition, clustering high dimensional data is complicated when relying on density measures. Graph-based (Hierarchical) clustering offers better semantic reasoning compared to the first two techniques. It enables a first level of semantic-based processing by exploiting the hierarchy and inter-cluster relations. In addition, Hierarchical Clustering is accurate but remains highly expensive computation wise. Finally, Conceptual Clustering presents two main advantages. Firstly, incremental algorithms exist and offer lower complexities for time and space. Secondly, these techniques (e.g., FCA) offer two levels of semantic reasoning: (i) handling formal concepts as nodes, and (ii) generating an ordered lattice of concepts (nodes). Finally, FCA is multi-modal, dynamic, and multi-source.

Table 9: Clustering Technique Comparison

Criterion	**Clustering Technique**			
	Prototype-based [19, 21]	Density-based [33]	Graph-based [5]	Conceptual [7, 30]
Multi-Modality	No	No	No	Yes
Multi-Source	No	No	No	Yes
Incremental Processing	No	No	No	Yes
Level of Human Intervention	High	Moderate	Moderate	Low
Predefined Cluster Number[5]	Yes	No	No	No

6 Conclusion & Future Work

Event Detection is an essential part of many applications/services from various application domains (e.g., social, medical, home/building management, energy management, navigation systems, industry and manufacturing). All these approaches are task-centric, and designed for a specific application domain/purpose. However, these works do not share a common re-usable backbone for event detection that can be instantiated/used in different contexts. In this paper, we propose a generic framework, the event virtual machine (eVM), for feature-centric event detection. Our approach allows to target events using an SQL-like query language (eQL), thus creating a specific instance for each use case using

[5] This criterion states if the final number of clusters is required prior to clustering.

the same framework. The detection part is based on Formal Concept Analysis (FCA), an incremental and dynamic clustering technique. We developed a prototype for testing purposes. The results show that our approach achieved high accuracy in most cases, especially when additional features (in addition to time and location) are considered. Results also proved that eVM is re-usable and multi-modal. As future work, we are investigating the detection of optimal granularities. We would also like to help improve the accuracy by automatically considering spatio-temporal distances between clusters and noise handling techniques. Finally, we want to extend the event language even more in order to test the extensibility, ease of integration, and evolution of the eVM framework based on the detection needs.

Acknowledgement

We thank Dr. Gilbert Tekli and Dr. Yudith Cardinale for their valuable feedback and input. We would also like to thank Anthony Nassar for his remarkable contribution in developing the mobile application used for the experimentation of this work.

References

1. Agarwal, Y., Balaji, B., Gupta, R., Lyles, J., Wei, M., Weng, T.: Occupancy-driven energy management for smart building automation. In: Proceedings of the 2nd ACM workshop on embedded sensing systems for energy-efficiency in building. pp. 1–6. ACM (2010)
2. Aggarwal, C.C.: Managing and mining sensor data. Springer Science & Business Media (2013)
3. Allan, J., Papka, R., Lavrenko, V.: On-line new event detection and tracking. In: Proceedings of the 21st annual international ACM SIGIR conference on Research and development in information retrieval. pp. 37–45. ACM (1998)
4. Bahrepour, M., Meratnia, N., Havinga, P.J.: Sensor fusion-based event detection in wireless sensor networks. In: Mobile and Ubiquitous Systems: Networking & Services, MobiQuitous, 2009. MobiQuitous' 09. 6th Annual International. pp. 1–8. IEEE (2009)
5. Berkhin, P.: A survey of clustering data mining techniques. In: Grouping multidimensional data, pp. 25–71. Springer (2006)
6. Buckman, A., Mayfield, M., BM Beck, S.: What is a smart building? Smart and Sustainable Built Environment 3(2), 92–109 (2014)
7. Burmeister, P.: Formal concept analysis with ConImp: Introduction to the basic features. Fachbereich Mathematik, Technische Universität Darmstadt (2003)
8. Cao, L., et al.: Image annotation within the context of personal photo collections using hierarchical event and scene models. IEEE Transactions on Multimedia 11(2), 208–219 (2009)
9. Chen, L., Roy, A.: Event detection from flickr data through wavelet-based spatial analysis. In: Conf. on Information and Knowl. Manag. pp. 523–532 (2009)
10. Choi, V.: Faster algorithms for constructing a concept (galois) lattice. Clustering Challenges in Biological Networks p. 169 (2006)

11. Chong, C.Y., Kumar, S.P.: Sensor networks: evolution, opportunities, and challenges. Proceedings of the IEEE 91(8), 1247–1256 (2003)
12. Compton, M., Barnaghi, P., Bermudez, L., GarcíA-Castro, R., Corcho, O., Cox, S., Graybeal, J., Hauswirth, M., Henson, C., Herzog, A., et al.: The ssn ontology of the w3c semantic sensor network incubator group. Web semantics: science, services and agents on the World Wide Web 17, 25–32 (2012)
13. Cooper, M., et al.: Temporal event clustering for digital photo collections. ACM Transac. on Multimedia Computing, Communic., and App. 1(3), 269–288 (2005)
14. Cui, J., et al.: Easyalbum: an interactive photo annotation system based on face clustering and re-ranking. In: Conf. on Human factors in computing systems. pp. 367–376. ACM (2007)
15. Doolin, D.M., Sitar, N.: Wireless sensors for wildfire monitoring. In: Smart Structures and Materials 2005: Sensors and Smart Structures Technologies for Civil, Mechanical, and Aerospace Systems. vol. 5765, pp. 477–485. International Society for Optics and Photonics (2005)
16. Ganter, B., Wille, R.: Formal concept analysis: mathematical foundations. Springer Science & Business Media (2012)
17. Hanlon, M., Anderson, R.: Real-time gait event detection using wearable sensors. Gait & posture 30(4), 523–527 (2009)
18. Holland, J.H.: Adaptation in natural and artificial systems: an introductory analysis with applications to biology, control, and artificial intelligence. U Michigan Press (1975)
19. Jain, A.K.: Data clustering: 50 years beyond k-means. Pattern recognition letters 31(8), 651–666 (2010)
20. Jasiewicz, J.M., Allum, J.H., Middleton, J.W., Barriskill, A., Condie, P., Purcell, B., Li, R.C.T.: Gait event detection using linear accelerometers or angular velocity transducers in able-bodied and spinal-cord injured individuals. Gait & posture 24(4), 502–509 (2006)
21. Koperski, K., Adhikary, J., Han, J.: Spatial data mining: progress and challenges survey paper. In: ACM SIGMOD Workshop on Research Issues on Data Mining and Knowledge Discovery. pp. 1–10 (1996)
22. Labeodan, T., De Bakker, C., Rosemann, A., Zeiler, W.: On the application of wireless sensors and actuators network in existing buildings for occupancy detection and occupancy-driven lighting control. Energy and Buildings 127, 75–83 (2016)
23. Lee, J., Bagheri, B., Kao, H.A.: A cyber-physical systems architecture for industry 4.0-based manufacturing systems. Manufacturing Letters 3, 18–23 (2015)
24. Leonardi, C., Cappellotto, A., Caraviello, M., Lepri, B., Antonelli, F.: Secondnose: an air quality mobile crowdsensing system. In: Proceedings of the 8th Nordic Conference on Human-Computer Interaction: Fun, Fast, Foundational. pp. 1051–1054. ACM (2014)
25. Li, S., Son, S.H., Stankovic, J.A.: Event detection services using data service middleware in distributed sensor networks. In: Information Processing in Sensor Networks. pp. 502–517. Springer (2003)
26. Mei, T., et al.: Probabilistic multimodality fusion for event based home photo clustering. In: Internat. Conf. on Multimedia and Expo. pp. 1757–1760 (2006)
27. Oeldorf-Hirsch, A., Sundar, S.S.: Social and technological motivations for online photo sharing. Journal of Broadcasting & Electronic Media 60(4), 624–642 (2016)
28. Papadopoulos, S., et al.: Cluster-based landmark and event detection for tagged photo collections. IEEE MultiMedia 18(1), 52–63 (2011)
29. Park, S.C., Park, M.K., Kang, M.G.: Super-resolution image reconstruction: a technical overview. IEEE signal processing magazine 20(3), 21–36 (2003)

30. Priss, U.: Formal concept analysis in information science. Arist 40(1), 521–543 (2006)
31. Quack, T., Leibe, B., Van Gool, L.: World-scale mining of objects and events from community photo collections. In: Internat. Conf. on Content-based image and video retrieval. pp. 47–56 (2008)
32. Raad, E.J., Chbeir, R.: Foto2events: From photos to event discovery and linking in online social networks. In: Internat. Conf. on Big Data and Cloud Computing. pp. 508–515. IEEE (2014)
33. Rehman, S.U., et al.: Dbscan: Past, present and future. In: Internat. Conf. on Applications of Digital Information and Web Technologies,. pp. 232–238 (2014)
34. Reuter, T., et al.: Reseed: social event detection dataset. In: Conf. on Multimedia Systems. pp. 35–40. ACM (2014)
35. Sakaki, T., Okazaki, M., Matsuo, Y.: Earthquake shakes twitter users: real-time event detection by social sensors. In: Proc. of Internat. Conf. on WWW. pp. 851–860. ACM (2010)
36. Sayyadi, H., et al.: Event detection and tracking in social streams. In: Icwsm (2009)
37. Sheba, S., Ramadoss, B., Balasundaram, S.: Event detection refinement using external tags for flickr collections. In: Intelligent Computing, Networking, and Informatics, pp. 369–375. Springer (2014)
38. Tan, P.N., et al.: Introduction to data mining. Pearson Education India (2006)
39. Van Der Merwe, D., Obiedkov, S., Kourie, D.: Addintent: A new incremental algorithm for constructing concept lattices. In: International Conference on Formal Concept Analysis. pp. 372–385. Springer (2004)
40. Wahyudi, W.A., Syazilawati, M.: Intelligent voice-based door access control system using adaptive-network-based fuzzy inference systems (anfis) for building security. Journal of Computer Science 3(5), 274–280 (2007)
41. Welch, J., Guilak, F., Baker, S.D.: A wireless ecg smart sensor for broad application in life threatening event detection. In: Engineering in Medicine and Biology Society, 2004. IEMBS'04. 26th Annual International Conference of the IEEE. vol. 2, pp. 3447–3449. IEEE (2004)
42. Wille, R.: Restructuring lattice theory: an approach based on hierarchies of concepts. In: Ordered sets, pp. 445–470. Springer (1982)
43. Wong, J.K., Li, H., Wang, S.: Intelligent building research: a review. Automation in construction 14(1), 143–159 (2005)
44. Wu, Y.H., Miller, H.J., Hung, M.C.: A gis-based decision support system for analysis of route choice in congested urban road networks. Journal of Geographical Systems 3(1), 3–24 (2001)
45. Yu, L., Wang, N., Meng, X.: Real-time forest fire detection with wireless sensor networks. In: Wireless Communications, Networking and Mobile Computing, 2005. Proceedings. 2005 International Conference on. vol. 2, pp. 1214–1217. IEEE (2005)
46. Zampolli, S., Elmi, I., Ahmed, F., Passini, M., Cardinali, G., Nicoletti, S., Dori, L.: An electronic nose based on solid state sensor arrays for low-cost indoor air quality monitoring applications. Sensors and Actuators B: Chemical 101(1-2), 39–46 (2004)