BibTeX:

Heterogeneous Stream Processing for Disaster Detection and Alarming

François Schnizler*, Thomas Liebig†, Shie Mannor*, Gustavo Souto†, Sebastian Bothe‡, Hendrik Stange‡
*Technion, Haifa, Israel
{francois, shie}@ee.technion.ac.il
†Technical University Dortmund, Artificial Intelligence Group, Dortmund, Germany
{thomas.liebig, gustavo.souto}@tu-dortmund.de
‡Fraunhofer IAIS, Knowledge Discovery, Sankt Augustin, Germany
{sebastian.bothe, hendrik.stange}@iais.fraunhofer.de

***Abstract*—We present a novel approach for event recognition in massive streams of heterogeneous data driven by privacy policies and big data event processing. New technologies in mobile computing combined with sensing infrastructures distributed in a city or country are generating massive, poly-structured spatio-temporal data. With a view on emergencies and disasters these various data sources enable early response and offer situative insights when integrated in an on-line incident recognition system.**

Our hereby presented system architecture integrates multi-faceted sensing and distributed event detection to identify, label and increase confidence in detected incidents. A higher flexibility than existing event detection approaches is achieved by combination of the data streams at a round table. At the round table the data flow adjusts itself during execution of the real-time detection system. This offers more robustness in case streams appear or disappear. The developed architecture is used in nation-wide and city-level incident recognition scenarios.

***Keywords*-massive heterogeneous data streams, event detection, real-time big data analytics**

I. Introduction

The increasing availability of massive heterogeneous streaming data for public organizations, governments and companies pushes their inclusion in incident recognition systems. Leveraging insights from these data streams offers a more detailed and real-time picture of traffic, communication, or social networks, to name a few, which still is a key challenge for early response and disaster management.

Recent disasters in Germany (floods, hurricanes etc.) and recurring critical states in the Dublin transport system show the relevance of an incident recognition system that makes use of the increasing number of available data sources. Involving civil service agencies such as the German Federal Office for Civil Protection (BBK) and Dublin City Council (DCC) poses additional requirements in terms of complexity of the event space as well as data security and privacy.

Our main goal is to design a distributed, privacy-preserving incident recognition system that is capable to scale with the number of data sources, to integrate various data characteristics and to detect incidents by fusing information derived from various data streams, handle uncertainty in the data, and label relevant events as enriched *incidents*.

These data sources comprise mobile network utilization data, social media streams (twitter), data on bus and train networks (see [1] for details). We also integrate expert feedback (crowdsourcing) for event labeling. Initial experiments on the first two data sources show that fusing different streams constructs a richer picture of reality [2], e.g. geo-referenced micro-postings (Twitter) and cellular mobile network utilization during the centennial flood in Germany.

Automated fusion of heterogeneous data streams is challenging due to the different data formats and spatio-temporal resolutions. Harmonization and preprocessing is required. Also, the sensor streams potentially observe different aspects of the same phenomenon. The granularity and latency of the data streams vary with the different capabilities and locations of the sensors. Thus one event happening at a fixed location and time is measured at various places and times. It is important to match the heterogeneous streams in order to profit from the multi-faceted evidence.

The paper at-hand tailors our approach for heterogeneous data streams processing for incident detection and alarming. Our approach consists of three steps reflected in three main system components: (1) event detection performed by the *Intelligent Sensor Agents (ISA components)*; (2) a *Round Table (RT component)* projects anomalies to an *ontology* of low level disaster events (incidents) by fusing and evaluating the derived multi-source information on a detected anomaly; and (3) *complex event processing component* for pattern matching and situation reconstruction. At the end recognized incidents are used for visualization and alarming to inform experts located at DCC or BBK.

The paper is structured as follows. The next section presents related work and necessary features of an event detection system. Section III describes our architecture, Section IV an example of its operation and Section V the implementation in a streams framework. Sections VI and VII detail two components. Finally, we provide a summary and discuss future steps.

II. Related Work

Our approach for real-time analysis of massive heterogeneous streaming data is inspired by the TechniBall system

[3], previous works on stream data analysis [2] and follows the Lambda architecture design principles for Big Data systems [4]. The data streams processed in this paper represent spatial time series. Based on these spatial time series, low-level events may be derived.

In general spatio-temporal data comes in a variety of forms and representations, depending on the domain, the observed phenomenon, and the observation method. In principle, there are three types of spatio-temporal data: *spatial time series*, *events*, and *trajectories*. A *spatial time series* consists of tuples $(attribute, object, time, location)$. An *event* of a particular type $event_i$ is triggered from a spatial time series under certain conditions and contains the tuples verifying these conditions $(event_i, object_n, time_n, location_n)$. A *trajectory* is a spatial time series for a particular $object_i$. It contains the location per time and is a series of tuples $(object_i, time_n, location_n)$.

Detecting events in spatio-temporal data is a widely investigated research area (see [5] for an overview). Previous work on spatial events detection [6] already extend the detection by a dynamic tracking of event clusters through space and time. In general functions for event detection can be classified using a former concept of raster-geography, namely *map-algebra* [7]. Both, raster geography and heterogeneous spatio-temporal data analysis consider data which is provided in multiple layers (i.e. one layer per data stream). Functions can be applied to one or multiple layers. Thus, spatial functions split into four groups: *local*, *focal*, *zonal* and *global* ones [7]. For heterogeneous data streams analysis, expressiveness of these four function types is important to derive low-level events (incidents), and for combining (e.g. aggregation, clustering, prediction etc.) low-level events to trigger high-level events (situations).

The exploitation of spatio-temporal event patterns is a major research field in mobility mining. Recently, pattern-graphs were introduced in [8], their pattern description is capable to express the temporal relations among various occurring events following the interval-calculus [9]. As an example the co-occurrence of two low-level events may trigger any high-level event. We extend their notion with spatial relations based on the region connection calculus [10] for detection of spatio-temporal patterns as part of the complex event processing component on situation reconstruction.

The complex event processing component can potentially be defined independently of the input streams. Possible frameworks are the event calculus [11], finite automaton [12] and other pattern matcher [13], [8] or even complex frameworks which allow application of local, focal, zonal and global functions e.g. [14], [3]. The requirements for spatio-temporal pattern matcher in our setting are:

- to operate in real time,
- to incorporate spatial [10] and temporal [9] relations
- to provide local, focal, zonal, and global [7] predicates on the attributes, and
- to pose arbitrary queries formed of these elements (regular language [15], Kleene closure [16]).

The recent challenge of the ACM conference on Distributed Event-Based Systems 2013 addressed some of these aspects by posing an event detection task in trajectories of soccer match players. The high sampling rate of the data pushed development of new data processing frameworks. Our approach uses this innovation, in particular the TechniBall solution [3] and its underlying streams framework [17]. The streams framework executes a predefined dataflow graph (defined in XML). The dataflow graph consists of descriptions of incoming streams and programmable processors which modify, filter and process the data streams. This generic scheme allows easy combination with (1) existing streaming query languages e.g. ESPER [18] or finite automaton for regular expression matching and (2) functions (aggregations, predictions, etc.). Limitations of the TechniBall system are:

- The TechniBall framework for event detection uses homogeneous data streams, in contrast this work utilizes heterogeneous data sources.
- The union of the different streams is expert-designed, fixed and non-flexible.

Others approaches focus on trajectories and project the derived mobility behavior into geographic semantic spaces [19] which is a similar concept to the abstraction done in a Round Table. However, this remains a semi-automated process.

To build an adaptable heterogeneous data streams aggregation system, information extracted from each data stream must be coded into a common language. This language is defined by an ontology, which is nothing new. For example, [20] also proposed an heterogeneous sensor aggregation platform using an ontology to query measurements. One key difference with our work is that we do not only describe a mechanism to query measurements or to communicate with sensors, but also a generic framework to fuse extracted information in order to detect relevant events. In other words, using existing frameworks typically requires the development of both an aggregation mechanism and of sensor specific processing algorithms, while ours only requires the latter. On the other hand, it is restricted to event detection while other frameworks can be used for other tasks.

III. Analysis workflow for handling massive heterogeneous streaming data

The components of the proposed system architecture are organized into four layers. The sensor streams are captured by the *input layer*, and then are passed to the specific *Intelligent Sensor Agent* (ISA) for cleaning, formatting, and event detection. The basic event data then enter the *Round Table* (RT) using a common abstraction model of the data in order to fuse information from various data sources.

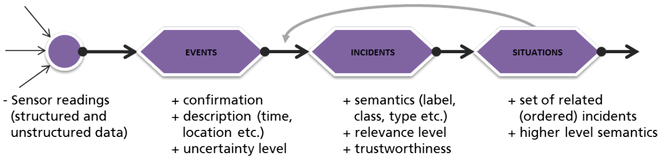
EVENTS
INCIDENTS
SITUATIONS
- Sensor readings (structured and unstructured data)
+ confirmation
+ description (time, location etc.)
+ uncertainty level
+ semantics (label, class, type etc.)
+ relevance level
+ trustworthiness
+ set of related (ordered) incidents
+ higher level semantics

Figure 1: Steps in situation reconstruction

The outcome are incidents which are fed into the *complex event detection* (CEP) component, which is responsible for aggregating, clustering, simulation or prediction on these incidents. The fusion of information within the Round Table and CEP takes place in the computation layer. The final results are disseminated to all interested parties via the *output layer*.

The *Round Table* and the *Intelligent Sensor Agents* are the two main components of the architecture developed in the INSIGHT project[1].

1) Events and incidents: For the sake of clarity Figure 1 illustrates a typical three-step knowledge chain of a situation reconstruction process. At the lowest level, we detect basic **events** in each individual data stream. Any redundancies or contradictions must, naturally, be resolved. If a RT is able to confirm and label the event using an ontology (add semantics to it) and assess its relevance, it becomes an **incident**.

Each incident is characterized by

- a location, the area affected by the incident, for example a polygon delimited by geographic coordinates;
- a time interval (potentially of length zero) when the event occurred;
- an event type, also called event semantics or labels, a classification of an event, for example flood, traffic jam or explosion.

Each incident is part of a **situation**; for our city-level and nation-wide use cases it may be a congestion, accident or more complex disaster situation. By grouping related incidents and ordering them correctly, separate situations can be identified. To support the relevance assessment and to put events into context all currently active situations must be known at the incident detection level. Detecting events and incidents are the central tasks of the RTs. Figure 2 outlines the general process flow.

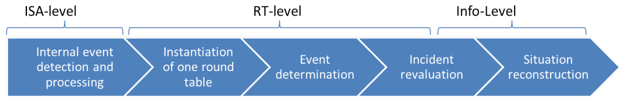

Figure 2: General process of sensor intelligence

2) Intelligent Sensor Agent: The ISA is a sensor-based *stream processing unit* integrated with a dedicated *analytics* and *learning engine* accompanied by a *local history*. Each ISA may be viewed as embodying an autonomous **Lambda architecture**, wherein the stream processor corresponds to the *speed layer*, the local history represents the *batch layer*, and the indexed knowledge computed from the history parallels the *serving layer*. For more details, see [21].

An ISA is adapted to a given *group* or *class* of input sensors and knows about sensor characteristics and embodies specific functions for data processing, cleansing, filtering and handling of missing values. Each ISA is also responsible for detecting events, that are indications of potential incidents.

In practice, this means that each ISA, or group thereof, is responsible for detecting known, suspicious, aberrant or interesting readings at the source. If such sensor readings are observed the detecting ISA suggests that the Round Table Manager (RTM) initiates a so-called ***Round Table***. Additionally, the ISA responds to queries from an already instantiated RT by integrating the most recent streaming data with its local store of historical knowledge (batch views).

Examples for low-level events that could be discovered in the ISA are: increasing numbers of Twitter messages, the reduction of density among moving objects, the mentioning of critical events in micro-postings or the decrease of traffic saturation.

The ISA may be deployed either locally or remotely, relative to the rest of the INSIGHT system:

- **Locally** on the same cluster as the RT and CEP components of the INSIGHT back-end. This naturally relies upon the data supplier's willingness to allow off-site access to their raw data, even over a secure connection.
- **Remotely** at the physical location (e.g. base station) from which the sensors are administered. If, for instance, the sensors provide highly sensitive data with high privacy constraints, it may be forbidden to access the data from an off-site location. Other data sources may not be allowed to be stored at the same location (e.g. billing and service data). Local ISAs share only abstract event descriptions with the INSIGHT back-end.

By adopting this design, we both delegate the initial detection of relevant or interesting events to the sensor level, and we decentralize the system's data, storing and processing it locally at the point where it is most needed and best understood. With this unique concept we achieve multiple benefits:

- sensor-specific, data-centric intelligence
- legal constraints satisfaction for each data source
- decentralization of data (e.g. for privacy reasons)
- individual storage strategies
- overall flexibility of the final system (e.g. extensibility).

The term "*sensor*" may in turn refer to an actual sensor, or a higher-level generalization from which our input data is drawn. For instance, we may not have direct access to every phone in a cellular network, but may receive

[1] www.insight-ict.eu/

cellular-sensitive network data instead. Due to the composite functionality of the ISA, it straddles the *input*, *preprocessing* and *computation* layers of the system architecture.

3) Round Table: Monitoring real world sensor data in a distributed and loosely coupled manner to detect relevant events requires joint analysis of various sources of data. With our approach to delegate the processing and local event detection to specialized ISA, we need a system to exchange information between these agents to come to a joint decision on events.

The RT is responsible for identifying, confirming and labeling events by fusing information from different ISAs, from crowdsourcing and/or from human experts. It is a multi-agent collaboration environment to validate, enrich and confirm events to become incidents in our terminology.

A RT is instantiated by the RTM whenever an basic event reported by an ISA must be investigated. The RT then monitors the situation by involving other ISAs to share their current state of information and to come to an agreement on whether an incident has happened or not and in the positive case associate a specific time interval, location and semantics to this incident.

The RT session may be envisioned as a "round table meeting of experts," where each expert has its own view on reality and knowledge. The main idea is that for each event detected by a single ISA we need to consult additional sources for complementary information and refinement of the current findings until we reach an agreement on whether or whether not a relevant incident has been detected. The process may include input from human analysts. As "Expert-in-the-loop" they guide algorithmic methods over all iteration steps (e.g. by labeling data or provide ground truth).

In general, objectives of a RT session are:

- achieving a higher spatial and/or temporal resolution,
- assigning a label to the event,
- increasing the confidence in the validity of the event,
- annotating a confirmed event with additional semantic information.

To achieve these goals we need to investigate any given event in a flow of events. For each anomaly one instance of the RT is initiated by the detecting agent. The structure of a RT is shown in Figure 3. Part of each RT is the moderator, the initiating ISA and invited participating. After initiation the RTM chooses and recruits a selection of other ISA (experts), which it believes may, together, be able to confirm an event or not.

To initiate the discussion, the requesting agent has to provide initial information to the group. This information may contain a time, location and type of event.

If the joint information provided by all agents is insufficient to decide on an event, the RT has three options for resolving the deadlock:

1) employ the system's crowdsourcing capabilities to solicit information from external actors, such as citizens who may be present in the vicinity of the event,
2) escalate the issue to a human expert, who is asked to manually identify and resolve the deadlock in the RT,
3) close the discussion without result.

The strategy selected very much depends on the use case and nature of the event at hand. Disasters with a nationwide magnitude may require the involvement of official actors whereas in a traffic scenario soliciting information from citizens may be constructive.

The general course of the RT process can be described by:

1) an ISA detects an anomaly;
2) the ISA requests the RTM to set up a RT session instance;
3) the RT selects data sources it believes are promising to evaluate the anomaly, the data sources are represented by their ISAs;
4) the initiator of the RT presents its finding to the RT;
5) the RT requests information from other participating agents;
6) all other participants explore their sensor data to verify an occurrence of an event showing the same characteristics as provided; this is an iterative process;
7) the RT receives and integrates information provided by each ISA;
8) if an event has been detected the most precise description of the anomaly is distilled from all data "on the table";
9) the RT sends information on the decision to the complex event processing.

This procedure is illustrated in Section IV and the RT inner workings are described in Section VI.

4) Ontology: Each ISA receives data items encoded using a stream specific format. At the RT level, information extracted from these data streams by each ISA are combined. The RT must therefore be able to understand these pieces of information.

To achieve this understanding, there are two possibilities: adapting the RT for every new type of ISA, or developing a common language and interface between the ISA and the RT. The first approach would contradict the modularity

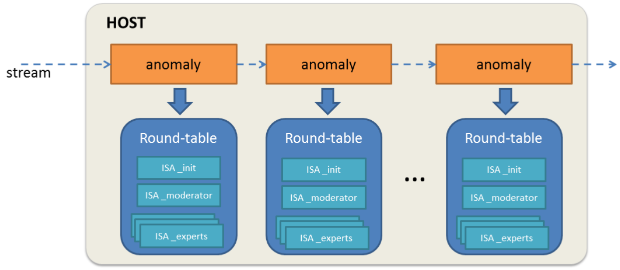

Figure 3: Structure and components of a Round Table

requirement of the INSIGHT system. We therefore select the other possibility. The ontology is the common language used by the RT, the RTM and the ISAs.

Each ISA therefore also acts as an interface, as a translator, between the raw data and higher level components of the system. The ISA encodes information extracted from the raw data into a language the rest of the INSIGHT system can understand. The words of this language are defined by the ontology.

Informally speaking, an ontology is a description of data. It describes the structure of the data, values allowed within the various elements of the structure, and a representation of the relationships between the allowed values. Such an ontology can be used in

- communication between systems, e.g. through web services, and
- communication within a system, e.g. in communication between stream processors in a stream processing framework, and possibly
- reasoning about the data, e.g. event detection, reconciling the output of multiple event detection engines.

The ontology developed for the INSIGHT system is subject for future improvements, including the definition of what are the *time*, *location* and *semantics*, relevant for the communication between the ISAs and the RT, and the reasoning.

5) Round Table Manager: The creation of the RT can be triggered by two elements. First, a RT can be created following an external query from a human expert (pull-based) or second, an ISA detects an abnormality which triggers the RT instantiation (push-based).

An event affecting a large area could result in events detected by multiple ISAs at distinct locations, potentially leading to the creation of multiple RTs. These RTs would then process the same event. These duplicates could be removed by the complex event processing constructing incidents, and some duplicates are probably unavoidable. However, processing the same event multiple times would consume resources without necessity, and should be avoided or mitigated.

To handle this type of issues, an overview of all existing RTs is necessary. This overview is taken care of by the RTM, which is responsible of the creation and the management of the RTs. The RTM is described in more details in Section VII.

IV. Operation illustration

To give an illustrative example of the RT process, assume that a mobile phone network sensor detects an abnormal reading at Cell $AB - 1234$, starting at 12:00 and not yet returned back to normal at 13:20. The sensor agent responsible for the mobile network sensor forwards this event to the RTM. The manager creates a RT session and adds the initiating agent (mobile phone network) and the agents responsible for the Twitter sensor to the list of participants. The initiator of the meeting shares the information on the anomaly with the RT. This information may be a 3-Tuple

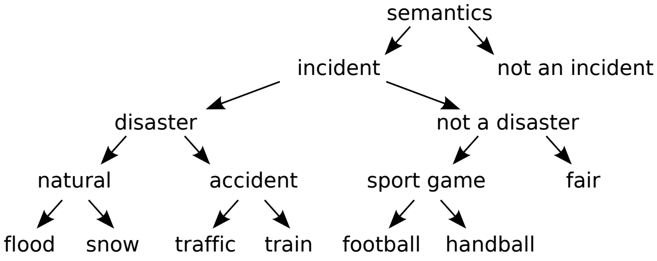

Figure 4: Illustration of an event label ontology

$$< location, time, \{< semantic, likelihood >\} >,$$

where the notation {...} denotes a list of elements. Each element of this tuple is encoded according to the ontology. For this example, we choose a concrete representation of the form:

$$< (lat, long, radius), [starttime, endtime], \\ \{eventType, likelihood\} > .$$

Let us also assume that eventType is an element of the ontology represented in Figure 4. This ontology contains catastrophic events such as “flood” or “car accident” and some other non catastrophic events such as ”football game”.

The requesting ISA could therefore report :

$$< (6.51, 9.87, 1500), [12:00, 13:20[, \{< incident, 1 >, \\ < not\_incident, 0.2 >\} > .$$

Based on this triple, the RT asks all participating ISAs to gather information about this incident, for example information about the same location, starting at 12:00. Since the event is not over yet, the RT does not specify an end time to this query, and the ISAs keep gathering and providing information to the RT.

The Twitter sensor agent receives the query and searches for relevant Tweets around the requested location. In this example it finds a large increase in the frequency of the hash tag “FC Köln” since 12:00. Therefore, it derives that with high probability the eventType is a soccer game.

This new information is now reported to the RT:

$$< (6, 9, 10000), [12:00, now], \{< soccer\_game, 1 >, \\ < handball, 0.2 >, < disaster, 0.1 >, \\ < not\_incident, 0.2 >\} > .$$

As there are only two sensor agents in the session, the RT can merge the two messages. For the semantics, a merge

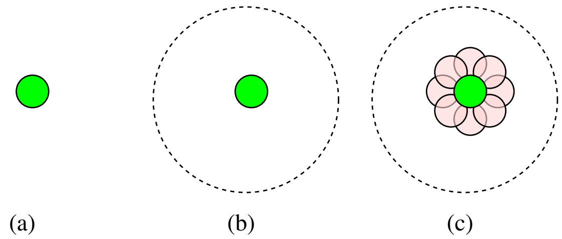

Figure 5: Illustration of an collection of information in the Round Table. The phone ISA first reports an anomaly in the location represented by the green circle (5a). The Twitter ISA searches for an event and identify a high number of football related tweets in the much larger dashed black circle (5b). To check whether the area affected is larger than the original area (green), the RT queries the phone agent for nearby locations, resulting in the phone agent providing information about the light red locations (5c).

could for example result in the following combination:

$$\begin{aligned}\{< soccer\_game, 1 >, < handball, 0.2 >,\\ < disaster, 0.1 >,\\ < not\_incident, 0.04 >\},\end{aligned}$$

and therefore to a more precise event label. The time is identical, so the merge is trivial. However, the locations do not match, as the Twitter ISA has a much coarser resolution than the phone ISA. According to the Twitter ISA, the area affected by the event could therefore be much larger than previously thought.

To resolve this uncertainty, the RT could in a second step query the phone ISA about cells next to the original cell (6.51,9.87,1500). This is illustrated in Figure 5. When the RT receives additional information about the extended area from the phone ISA, it can decide to extend the area of the original event, or not.

When a sufficient confidence is reached by the RT, the RTM outputs the result for further processing and/or display to human operators.

The RT monitors the incident until it ends, providing updates about the affected locations and signaling its termination.

V. Implementation of the system

This section focusses on the implementation of the INSIGHT system. In Section V-A, we describe the relationships between the different components of the RT. Finally Section V-B considers the implementation of the system in a streaming framework.

Different frameworks and hardware can be used to build such a system (see e.g. [21]). The focus of this section is not on discussing these technologies, but rather on implementing the system using such technologies.

A. Interface description

Communication between elements of the system is centered around two types of messages, the queries generated by the RT and the computations of the ISA. This includes both the original events detected independently of any RT query and the answers computed because of opened queries.

This section first describes the instantaneous relationships between the ISA, the RT and the data they exchange. Java interfaces allowing to create and manage these relationships are provided publicly to facilitate usage of the hereby presented system.

An ISA is characterized by its ID and its description. An ISA can be assigned to RTs by the RTM. Each RT is characterized by an ID. A RT sends a spatio-temporal query to each ISA, that is a location and a start time. This query is currently the same for all ISAs, but this can be modified easily.

The result of the computations of the ISA is formatted as a *Round Table Data* (*RTData*). A Round Table Data is composed of the following elements:

- *RTDataID*, a unique identifier,
- *timeStamp*, the time the RTData was generated,
- *timeInterval*, the time interval the RTData applies to,
- *location*, the area the round data applies to,
- the likelihood of different *eventTypes* for this particular location and time interval.

Both the ID and the time stamps compose the key of a RTData. This allows an agent to update a previous RTData.

RTData are communicated by the ISA to the RTs. There are two possible relationships between and agent, a RT and a RTData. The RTData either was at the origin of the RT or was communicated to the RT based on a query to the agent.

In a distributed system, the delays between the generation of RTData and their reception by the RT / RTM may be significant. To monitor the system, the time stamp of the reception of a RTData must be recorded.

B. Implementation in a streaming framework

The INSIGHT system deals with massive amounts of data. Moreover, ISAs can be physically located in a different location than the main INSIGHT architecture. The INSIGHT system is therefore designed with a streaming architecture in mind. This section considers the implementation of the RT using the TUD streams framework [17]. Adapting the content of this section to another streaming framework is straightforward.

The streams framework is based on four abstract elements:

- *containers* are environments where stream sources, services and processes exist and are executed,
- *streams* or *queues* provide access to a sequence of data items,
- *processes* are active elements that read from a stream and execute a set of processors for each item obtained from the stream,

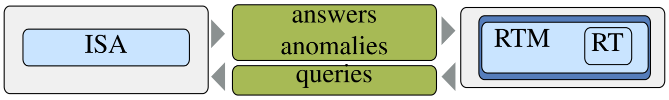

Figure 6: The Round Table (RT) and Round Table Manager (RTM) are implemented in the TUD streams framework as the same processor, here represented with the communication channels between the RTM / RTs container and the ISA containers.

- *services* are essentially simple sets of functions that are exported to be accessible from anywhere at anytime.

The interplay between these elements is illustrated in [17].

Since ISAs operate independently from each other and possibly in different locations and therefore on different machines, ISAs are likely to run in different containers. However, different ISAs can also be implemented inside a common container, for example if they operate on data collected at the same location.

RTs and the RTM are closely related, since the latter creates and monitors instances of the former. Therefore they operate in the same container. Several organizations are possible within that container. We placed the RTs and the RTM in the same processor (Figure 6, right). Using the same processor facilitates the communication (no need to use services), but is harder to modify in case using several machines becomes necessary at a later stage. See [22] for alternatives.

Setting up the communications between the ISAs and the RT using TUD streams involves two decisions: encoding the messages (anomalies, queries and answers) exchanged by the different entities and transmitting them. Since the content of the messages is described by the ontology, formatting the messages reduces to selecting a data-interchange format. We use JSON. We transmit anomalies and answers using the same channel. Both are emitted by the ISA and mostly consist of a RTData item. The only difference is the receiver: the RTM or the RT. They can easily use the same channel if, for example, anomalies are identified by the fact their RT ID is null. The second decision is whether a *stream* or a *service* is used for each channel. We expect a high volumes of all three message types. We therefore implement them all as streams, as illustrated in Figure 6.

VI. Round Table formalization

A RT receives many information from the ISAs, and must evaluate the occurrence of an incident. In this section, we formalize the task of the RT as an inference problem and propose a solution. Figure 7 illustrates the formalization.

Let a random variable $\mathcal{Z}$ be the label of an unknown interesting incident. Admissible values of $\mathcal{Z}$ are the nodes of a tree structure, defined by an ontology (see Figure 4 for an example).

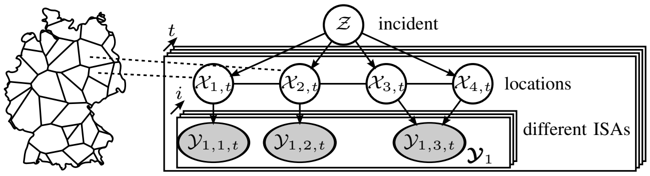

Figure 7: Schematic representation of the Round Table problem formulation. An event (of label $\mathcal{Z}$) can affect a set of locations L. Each variable $\mathcal{X}_{l,t}$ indicates whether location l is affected at time t. Different heterogeneous sensor networks $\boldsymbol{\mathcal{Y}}_i = \{\mathcal{Y}_{i,1,t}, \mathcal{Y}_{i,2,t}, \ldots\}$ monitor the state of the locations over time. Grayed nodes are observed measurements, white ones are unobserved variables. Each inner rectangle represents one ISA processing one sensor stream $\boldsymbol{\mathcal{Y}}_i$, outer rectangles different time steps. Arrows between $\mathcal{X}$ and $\mathcal{Y}$ represent the geometric relationship between the sensor specific location and the incident specific location. For example, a Twitter ISA might have a coarser geographical resolution than physical sensor ISAs.

The area being monitored (for example, Dublin or Germany) is prepartitioned into a set of locations L. For any given time t and a specific location $l \in L$ the random binary variable $\mathcal{X}_{l,t}$ indicates whether l is affect by an event at time t. $\boldsymbol{\mathcal{X}} = \{\boldsymbol{\mathcal{X}}_t\}_{t \in T}$ denotes the set of all elements indexed by time t during monitoring time T and $\boldsymbol{\mathcal{X}}_t = \{\mathcal{X}_{l,t}\}_{l \in L}$. $P(\boldsymbol{\mathcal{X}}|\mathcal{Z})$ encodes the spatio-temporal characteristics of each event. For example, a flooding incident is likely to affect a larger area while a football match is more localized.

Each heterogeneous data stream $\boldsymbol{\mathcal{Y}}_i$ is processed by an ISA and interpreted as a set of random variables $\{\mathcal{Y}_{i,j,t}\}$, and each variable $\mathcal{Y}_{i,j,t}$ regroups the measurements related to one or several locations, denoted by $\boldsymbol{\mathcal{P}}_{\mathcal{Y}_{i,j,t}}$.

Furthermore, we assume that the locations associated to sensor measurements inside a specific stream are non-overlapping. In other words $\boldsymbol{\mathcal{P}}_{\mathcal{Y}_{i,j,t}} \cap \boldsymbol{\mathcal{P}}_{\mathcal{Y}_{i,j',t}} = \emptyset \; \forall j \neq j'$. The idea behind the ISA is that it understands the nature of the sensor network and is capable of not only detecting potential events (anomalies) but also to prepare the data so that it is consumable by the RT. In particular, it can enforce the non-overlapping location assumption, although we believe it is not very strong. Indeed, physical sensors are typically associated to well-defined geographic locations, for example the location of the sensor, the position of the public transport vehicle equipped with GPS or the area serviced by a mobile phone cell tower. Geo-localized measurements such as tweets can either be handled individually as point wise measurements, or handled as a group of measurements based on proximity, leading to the definition of cells.

The goal of the RT is to infer the values of the variables $\mathcal{Z} \cup \boldsymbol{\mathcal{X}}$, based on the measurements $\{\boldsymbol{\mathcal{Y}}_i\}_i$. To perform this inference, the RT has access to the following elements, made available by each activated ISA: $P(\mathcal{Y}_{i,j,t}|\mathcal{Z}, \boldsymbol{\mathcal{P}}_{\mathcal{Y}_{i,j,t}})$.

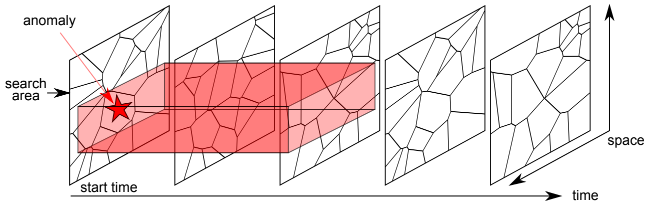

Figure 8: A bounding box is constructed around an anomaly to maximize the likelihood that the area inside the box is affected by an event. Boxes too small are discarded as noise.

1) objective function: Two objective functions can be considered. The first one is the *maximum likelihood* (ML) explanation of the observations $\boldsymbol{y}$, that is

$$\{\hat{z};\hat{\boldsymbol{x}}\} = \arg\max_{z;\boldsymbol{x}} P(\boldsymbol{\mathcal{Y}}=\boldsymbol{y}|\mathcal{Z}=z;\boldsymbol{\mathcal{X}}=\boldsymbol{x}) \ , \qquad (1)$$

under some constraints on the configurations of the variables $\boldsymbol{\mathcal{X}}$ to ensure realistic instantiation. These constraints could for example forbid the consideration of a set of affected locations that are not connected for a single anomaly.

The second is the *maximum a posteriori* (MAP) estimate of the unknown variables, that is

$$\{\hat{z_{MAP}};\hat{\boldsymbol{x}}_{MAP}\} = \arg\max_{z;\boldsymbol{x}} P(\mathcal{Z}=z;\boldsymbol{\mathcal{X}}=\boldsymbol{x}|\boldsymbol{\mathcal{Y}}=\boldsymbol{y}) \ . \qquad (2)$$

However, MAP requires the joint probability of the different events and of the area affected, $P(\mathcal{Z}=z;\boldsymbol{\mathcal{X}}=\boldsymbol{x})$. In the context of disaster monitoring, this quantity is typically not available, hence why we consider maximum likelihood inference and not MAP. In other contexts, the RT can easily be updated to perform MAP inference using approximate inference algorithms (see for example [23]).

2) ML inference: The complexity of this algorithm depends on the constraints on the variables $\boldsymbol{\mathcal{X}}$. Exploring all possibilities and associated tradeoffs are outside the scope of this paper. We describe a simple optimization scheme.

As our constraint, we restrict the shape of the spatio-temporal affected area to a rectangular bounding box. This is illustrated in Figure 8. Any variable $\mathcal{X}_{i,j,t}$ whose area is inside (or intersects) the box takes the value 'affected by the incident', any other variable is 'not affected by the incident'.

Additionally, we want any solution $\hat{z};\hat{\boldsymbol{x}}$ to contain the anomaly. In other words, the variable(s) $\boldsymbol{\mathcal{X}}_{i,t}$ associated to the location of the initial anomaly must be affected by the event. We therefore initialize the inference algorithm by only considering this area and time is affected by the event. We also initialize the eventType using the likelihood provided with the original anomaly.

We then use a greedy approach to gradually increase this area and refine the eventType based on additional information provided by the ISA. More specifically, we use alternating optimization to iteratively refine the location, time and eventType associated to the incident until we reach a local optimum of the function $P(\boldsymbol{\mathcal{Y}}=\boldsymbol{y}|\mathcal{Z}=z;\boldsymbol{\mathcal{X}}=\boldsymbol{x})$.

For a given eventType, the current bounding box is compared to candidate boxes in terms of their likelihood. Candidate boxes are generated by moving one side of the current box perpendicularly to itself in order to include or exclude a single RTData.

Within a given spatio-temporal box, inferring a label is easy. Let us first assume that all ISAs provide the likelihood for the same set of eventTypes. Given our assumption that observations provided by different ISAs and/or about different cells and/or different times are independent, likelihoods are multiplicative:

$$P(y,y'|\mathcal{Z}=z;\boldsymbol{\mathcal{X}}=\boldsymbol{x}) = P(y|\mathcal{Z}=z;\boldsymbol{\mathcal{X}}=\boldsymbol{x}) \\ P(y'|\mathcal{Z}=z;\boldsymbol{\mathcal{X}}=\boldsymbol{x}) \ .$$

Computing a likelihood for all eventTypes is therefore trivial. Let us now assume that RTData items a and b provide likelihoods for different sets of eventTypes S_a and S_b. EventTypes in $S_a^b = S_a \backslash S_b$ are attributed the same likelihood as their respective parents, and vice versa. This make any EventType in S_a^b as likely as its siblings and parent and therefore does not provide any new information. However, this allows to combine the two RTData items.

This algorithm has the advantage to be highly scalable. It also naturally directs the acquisition of information from the ISAs. To deal with the unbounded network size, we collect information in a spatial bounding box around the area initially investigated, from the initial time of the anomaly until the current time or the end of the incident. The spatial limit of this network are initialized slightly larger than the area where the anomaly is detected. These limit are increased when the estimation of the area affected reaches these limits.

VII. Round Table Manager details

The RTM has two primary tasks: dealing with basic events and monitoring RTs.

Anomalies detected by any ISA are handled by the RTM. It can (a) send the anomaly to an existing RT, so that the RT includes it in its current investigation, (b) instantiate a new RT to study the anomaly, (c) ignore the anomaly. Option (a) is triggered if the anomaly is close to the area investigated by the RT. There are many possible quantizations of 'closeness'. We propose to use a specified percentage of the area being investigated. Option (c) is only considered if too many events are already investigated for the computing resources available. In that case, the RTM prioritizes the investigation of measurements with the highest likelihood of an event. When a RT is created, the RTM add ISAs to this RT. At the moment, we include all available ISAs.

As mentioned before, the RTM also merges RTs that investigate similar events. An large event can generate anomalies at distinct locations, potentially leading to the creation of multiple RTs. To avoid duplicates, the RTM merge RTs investigating similar locations, that is RTs whose bounding box spatial locations A and B are such that $area(A \cup B) < \alpha\ area(A \cap B)$, where α is a user specified parameter.

The RTM is also responsible for monitoring instances of RTs so that the system keeps detecting events in real-time, that is, so that RTs do not require more computing resources than available. When the computing load reaches the acceptable limit, the RTM must reduce it. Several options are available: (a) reduce the history of RTs or (b) stop some RTs. Reducing the history consists in closing a RT and creating a new one that will operates only on RTData about measurements more recent than a user specified time interval, say 15 minutes. In other words, the new RT will monitor the situation starting 15 minutes ago. Limiting the lifespan of RTs can also be used to generate shorter events in time, in case this is preferred for the operation of the complex event monitoring engine. Killing existing RTs is a more drastic measure: some anomalies won't be investigated completely. This is a trade-off between monitoring ongoing events and investigating new ones. As for new anomaly investigation, we suggest prioritizing anomalies associated to the highest likelihood of an event.

VIII. Summary and Discussion

We presented a system for incident recognition from massive heterogeneous spatio-temporal data streams and provided an example for its operation. In each **Intelligent Sensor Agent** the data from all sensors of a specific type is preprocessed and low level events (anomalies) are derived. Our approach poses no assumptions on the location for this preprocessing. It could be in-situ at the sensor level or on a centralized computing node. The functions in this analysis phase may be local, focal and global ones. This could be signal processing functions (filters, discretization and thresholds) but also projections, classification, prediction, clustering and their combinations. Examples for low-level events (anomalies) that are discovered in the intelligent sensor agent are: the increase of the number of twitter messages, the reduction of density among moving objects, the stagnation of a water level sensor or the decrease of traffic flux (vehicles per time) or traffic speed at some location. In case of sparsely observed phenomenon, the intelligent sensor may impute missing data values. The ISA is also able to answer queries about specific location and time interval. This capability is used by the next component to obtain information about neighboring locations and/or time steps, or about events detected by another ISA.

A **Round Table** is instantiated whenever an anomaly is detected by an ISA. The Task of the RT is to map the anomaly to an event ontology and to refine its location and time interval. The heterogeneous data from multiple ISAs, obtained by querying them, is used for evaluation and interference of the eventType and reduction of uncertainty. The RT connects heterogeneous data sources by joining their ISAs. Some of them have to be invited and are included on request, e.g., human expert feedback (crowdsourcing). Others require adjustment of their parameters, e.g., the focus of a twitter sensor.

These two components fuse information from the input data streams into a single streams of incidents, defined by the ontology, and therefore not depending on the format of the original heterogeneous inputs of the system. Then, a typical **Complex Event Processing** component, not discussed in this paper, processes the single stream of detected incidents and combines them to high level situations which are of interest for the end user.

IX. Future Work

We will use the hereby presented INSIGHT system and develop ISAs for the available data sources. The incorporation of forecasting models as well as improvement of the ontology are also important paths to follow.

The RTM can also be enhanced. In particular, the RTM can learn from past incidents and learn which ISAs should be called to a RT to investigate a particular event.

Acknowledgment

This work is funded by the EU FP7 INSIGHT[2] project (Intelligent Synthesis and Real-time Response using Massive Streaming of Heterogeneous Data), 318225.

References

[1] T. Liebig and K. Morik, "Report on end-user requirements, test data, and on prototype definitions," TU Dortmund and Insight Consortium Members, Tech. Rep. FP7-318225 D5.1, August 2013.

[2] G. Fuchs, N. Andrienko, G. Andrienko, S. Bothe, and H. Stange, "Tracing the german centennial flood in the stream of tweets: First lessons learned," 2013.

[3] A. Gal, S. Keren, M. Sondak, M. Weidlich, H. Blom, and C. Bockermann, "Grand challenge: The techniball system," in *Proceedings of the 7th ACM International Conference on Distributed Event-based Systems*, ser. DEBS '13. New York, NY, USA: ACM, 2013, pp. 319–324.

[4] N. Marz, *Big data : principles and best practices of scalable realtime data systems*. O'Reilly Media, 2013. [Online]. Available: http://www.amazon.de/Big-Data-Principles-Practices-Scalable/dp/1617290343

[5] C. C. Aggarwal, *Outlier Detection*. New York, NY, USA: Springer, 2013.

[6] N. Andrienko, G. Andrienko, G. Fuchs, and H. Stange, "Detecting and tracking dynamic clusters of spatial events," in *VAST Challenge, 2014 IEEE*. IEEE, 2014.

[2] www.insight-ict.eu/

[7] J. K. Berry, “Gis modeling and analysis,” in *Manual of Geographic Information Systems*, M. Madden, A. S. for Photogrammetry, and R. Sensing, Eds. American Society for Photogrammetry and Remote Sensing, 2009, pp. 527–585. [Online]. Available: http://books.google.de/books?id=ek-IQAAACAAJ

[8] S. Peter, F. Höppner, and M. R. Berthold, “Learning pattern graphs for multivariate temporal pattern retrieval,” in *Advances in Intelligent Data Analysis XI*, ser. Lecture Notes in Computer Science, J. Hollmen, F. Klawonn, and A. Tucker, Eds. Springer Berlin Heidelberg, 2012, vol. 7619, pp. 264–275.

[9] J. F. Allen, “Maintaining knowledge about temporal intervals,” *Commun. ACM*, vol. 26, no. 11, pp. 832–843, Nov. 1983. [Online]. Available: http://doi.acm.org/10.1145/182.358434

[10] D. A. Randell, Z. Cui, and A. G. Cohn, “A spatial logic based on regions and connection,” in *KR*, 1992, pp. 165–176.

[11] A. Skarlatidis, G. Paliouras, G. A. Vouros, and A. Artikis, “Probabilistic event calculus based on markov logic networks.” in *RuleML America*, ser. Lecture Notes in Computer Science, F. Olken, M. Palmirani, and D. Sottara, Eds., vol. 7018. Springer, 2011, pp. 155–170.

[12] S. Florescu, C. Körner, M. Mock, and M. May, “Efficient mobility pattern stream matching on mobile devices,” in *Proc. of the Ubiquitous Data Mining Workshop (UDM 2012)*, 2012, pp. 23–27.

[13] A. Demers, J. Gehrke, B. Panda, M. Riedewald, V. Sharma, and W. White, “Cayuga : A General Purpose Event Monitoring System,” *Publish*, pp. 412–422, 2007.

[14] M. A. Sakr and R. H. Güting, “Spatiotemporal pattern queries,” *GeoInformatica*, vol. 15, no. 3, pp. 497–540, 2011.

[15] C. du Mouza, P. Rigaux, and M. Scholl, “Efficient evaluation of parameterized pattern queries,” in *CIKM*, 2005, pp. 728–735.

[16] D. Gyllstrom, J. Agrawal, Y. Diao, and N. Immerman, “On supporting kleene closure over event streams,” in *ICDE*, 2008, pp. 1391–1393.

[17] C. Bockermann and H. Blom, “The streams framework,” TU Dortmund University, Tech. Rep. 5, 12 2012. [Online]. Available: http://jwall.org/streams/tr.pdf [Last accessed: 28 November 2013]

[18] F. Marinescu, “Esper: High volume event stream processing and correlation in java,” Online article, July 2006. [Online]. Available: http://www.infoq.com/news/Esper–ESP-CEP

[19] N. Andrienko, G. Andrienko, and G. Fuchs, “Analysis of mobility behaviors in geographic and semantic spaces,” in *VAST Challenge, 2014 IEEE*. IEEE, 2014.

[20] C.-H. Lee, D. Birch, C. Wu, D. Silva, O. Tsinalis, Y. Li, S. Yan, M. Ghanem, and Y. Guo, “Building a generic platform for big sensor data application,” in *Big Data, 2013 IEEE International Conference on*. IEEE, 2013, pp. 94–102.

[21] B. Gorman, “Insight system requirements spec, standards and guidelines for development and architecture,” IBM Research - Ireland, BBK and Insight Consortium Members, Tech. Rep. FP7-318225 D2.1, August 2013.

[22] F. Schnitzler and S. Mannor, “Report on the development of online analysis and fusing techniques for sensor data,” Technion and Insight Consortium Members, Tech. Rep. FP7-318225 D4.1, August 2014.

[23] A. Darwiche, *Modeling and Reasoning with Bayesian Networks*. New York, NY, USA: Cambridge University Press, 2009.