UNIVERSITÀ DEGLI STUDI DI MILANO - BICOCCA

Facoltà di Scienze Matematiche, Fisiche e Naturali

Corso di Dottorato in Informatica

Criticality assessment of terrorism related events at different time scales

TENSOR clusTEriNg terroriSm actiOn pRediction

A dissertation presented by
RAUL SORMANI

Submitted in partial fulfillment of the requirements for the degree of
DOCTOR of PHILOSOPHY

Supervisor: Prof. Francesco Archetti
Supervisor: Prof. Francesco Tisato
Tutor: Prof. Lucia Pomello

April 2016

Abstract

Law Enforcement Agencies (LEAs) are nowadays taking advantage of a wide range of information and intelligence sources (e.g., human intelligence (HUMINT), open source intelligence (OSINT), image analysis (IMINT)) to anticipate potential terroristic actions. Urban environments are nowadays associated with a wide range of vulnerabilities, which create fertile ground for terrorists planning actions against assets and/or citizens. These vulnerabilities stem from the characteristics of the urban environment (e.g., presence of civilians, availability of many and diverse physical infrastructures, complex social/cultural/governmental interactions, high value targets, etc.) have been repeatedly manifested as part of major terrorist attacks, which took place in some of the world's most important cities (e.g., New York, London, and Madrid). The mitigation of security concerns in the urban environment is therefore a top priority in the social and political agendas of cities. ICT technologies provide help in this direction, for example through surveillance of urban areas, using the proliferating number of low-cost multi-purpose sensors in conjunction with emerging Big Data processing techniques for analyzing them.

The thesis illustrates the TENSOR (clusTEriNg terroriSm actiOn pRediction) framework, a near real-time reasoning framework for early identification and prediction of potential threat situations (e.g. terrorist actions).

The main objective of TENSOR is to show how patterns of strategic terroristic behaviors, identified analyzing large longitudinal data sets, can be linked to short term activity patterns identified analyzing feeds by "usual" surveillance technologies and that this fusion allows a better detection of terrorist threats.

The framework consists of three different modules with the aim of collecting and processing information of the surrounding environment from a variety of sources including physical sensors (e.g. surveillance cameras) and "virtual" sensors (e.g. police officers, citizens). The proposed TENSOR framework processes information sources at different abstraction levels (e.g. sensor information, police inputs, external semantic crafted data sources) and, thru the proposed layered architecture, simulates the three main expert user roles (i.e. operational, tactical and strategic user roles), as indicated in the intelligence analysis domain literature. The framework transforms all the sensors gathered data into symbolic events of interest following a generic scenario-agnostic semantics for terrorist attacks described in literature as terrorist indicators. Thru different reasoning and fusion techniques, the framework proactively detects threats and depicts the situation in near real-time.

The framework results have been tested and validated in the European project FP7 PROACTIVE.

Publications

Journals

Castelli, M., **Sormani**, R., Trujillo, L., & Popovic, A., "Predicting Per Capita Violent Crimes in Urban Areas: an Artificial Intelligence Approach", Journal of Ambient Intelligence and Humanized Computing (in press).

Sormani, R., Soldatos, J., Vassilaras, S., Tisato, F., & Giordani, I., "A Serious Game empowering the Prediction of Potential Terroristic Actions", Journal of Policing, Intelligence and Counter Terrorism, Vol. 11, Iss. 1, 2016.

Petris, S., Georgoulis, C., Soldatos, J., Giordani, I., **Sormani**, R., & Djordjevic, D. (2014). Predicting Terroristic Attacks in Urban Environments: An Internet-of-Things Approach. International Journal of Security and Its Applications, 8(4), 195-218.

Conferences

Archetti, F., Djordjevic, D., Giordani, I., **Sormani**, R., & Tisato, F. (2014, October). A reasoning approach for modelling and predicting terroristic attacks in urban environments. In Security Technology (ICCST), 2014 International Carnahan Conference on (pp. 1-6). IEEE.

Contents

Abstract II

Publications III

Contents IV

1. Introduction 1

1.1. Topic relevance 1

1.2. Contextualization and contributions 2

1.3. Thesis Outline 6

2. Application domain 7

2.1. Users classification 8

2.2. Alert level 9

2.3. Threat type overview 9

2.3.1. Non-information related terrorism 10

2.3.2. Information related terrorism 10

2.4. A general architecture used in urban security domain 11

2.4.1. JDL model 11

2.5. The data available in security domain 14

2.5.1. Security databases 14

2.5.2. Minorities at Risk Organizational Behaviour (MAROB) 16

2.5.3. Global Terrorist Database 17

2.5.4. Gaming data 22

3. State of the art 24

3.1. AI threat detection approaches 24

3.1.1. Information Extraction 24

3.1.2. Machine Learning 24

3.2. Investigated approaches 28

3.2.1. HMM 28

3.2.2. Clustering 30

3.2.3. CEP 33

3.3. Software tools and technologies 34

3.3.1. Big Data Mining 35

3.3.2. Machine Learning for Big and Streaming Data 37

3.3.3. Evaluating Machine Learning for Big and Streaming data 41

4. TENSOR: the proposed architecture 45

4.1. Framework overview 45

4.1.1. Sensor data collection 46

4.1.2. User requirements 47

4.1.3. TENSOR Framework architecture 49

4.1.4. STR - MTR - LTR in the JDL model 51

5. Short Term Reasoning layer 52

5.1. Hidden Markov Model 52

5.2. Description of HMMs approach 53

5.3. Mathematical definition 55

5.4. Assumption 56

5.5. Alert levels 58

5.6. Micro-environments 59

5.6.1. Micro-models 60

5.6.2. Training the micro-models 61

5.6.3. Exploiting historical series 62

5.7. Gaming data 63

6. Long Term Reasoning layer 68

6.1. Mapping between GTD data and event dictionary 69

6.2. Prediction activity 70

6.2.1. Spectral clustering vs k-means 72

6.2.2. Experimental settings and performance evaluation 73

6.3. Models update 76

6.3.1. User-in-the-loop approach 76

6.3.2. Micro-environments Self-Adaptive System 77

6.3.3. Proposed model update approach 79

7. Medium Term reasoning layer 80

7.1. Decision Support System overview 80

7.2. The developed approach 82

7.3. Improvements for the MTR module 89

8. Conclusion 90

Acknowledgements 91

References 91

Sitography 98

List of Figures

Figure 1 JDL Layers... 12
Figure 2 MAROB attacks for year grouped by country. 17
Figure 3 Trend attacks in the period 1970- 2014. 17
Figure 4 Number of instances for each target type. 18
Figure 5 Attacks per Year grouped by target type. 19
Figure 6 Number of instances for each action performed. 21
Figure 7 Attacks for year grouped by actions performed. 21
Figure 8 Knowledge value of complex events for an online event stream processing system [Fülöp et al., 2012]... 34
Figure 9 Layered solution approach. 50
Figure 10 JDL Layers and STR, MTR and LTR 51
Figure 11 Short Term Reasoning. 52
Figure 12 HMM transition states. 54
Figure 13 HMM observation probability. 54
Figure 14 Trend of threat probability without observation. 56
Figure 15 High Alert level with one Low relevance observation. 57
Figure 16 High Alert level with one Low relevance observation. 57
Figure 17 High Alert level with many Low relevance observations. 57
Figure 18 Low Alert level with many Low relevance observations. 58
Figure 19 Change of alert level: from Low to High alert level. 58
Figure 20 Micro-environment structure. 60
Figure 21 Micro-model structure. 61
Figure 22 Micro-models training. 62
Figure 23 Validation Scenario Map. 65
Figure 24 Long Term Reasoning (red lines). 68
Figure 25 Long Term Reasoning module. 71
Figure 26 Spectral clustering vs K-means (Silhouette index). 73
Figure 27 Spectral clustering vs K-means (Calinski Harabasz index). 73
Figure 28 F-measure spectral clustering. 75
Figure 29 Accuracy spectral clustering. 75
Figure 30 User-in-the-loop learning approach. 76
Figure 31 Medium Term Reasoning (orange lines). 80
Figure 32 Functional CEP system architecture. 84
Figure 33 Medium Term Reasoner GUI. 88

List of Tables

Table 1 UK and US Threat levels. 9
Table 2 Short description of terrorist datasets. 15
Table 3 Data processing engines for Hadoop [Landset et al., 2015]. 37
Table 4 Overview of machine learning toolkits [Landset et al., 2015]. 40
Table 5 Classification of Terrorist Indicators. 46
Table 6 Probability of actions generating events. 64
Table 7 Symbolic Event Example. 65
Table 8 Traditional evaluation metric. 66
Table 9 User’s evaluation metrics. 67
Table 10 Physical environment mapping. 69
Table 11 GTD action features mapping. 70
Table 12 Types of rules from a functional perspective as a CEP engine. 85
Table 13 MTR update Alert Level (AL). 86
Table 14 Example of spatial constraints rules. 88

1. Introduction

Terroristic attacks have spiked dramatically on European soil, both in terms of frequency and loss of human life making for an increasing awareness of the need of novel approaches to fight radicalization and terrorism.

The International NY Times, March 30 2016 pointing out the scale up of recent attacks wrote that "*for years authorities have discounted small attacks as isolated random acts*".

Quite to the contrary in the last 2 months investigations have shown that terrorist cells, largely ISIS controlled, have been revving up their machinery at least since early 2014 carrying out smaller attacks, while the devastating ones were in the making.

"*It's a factory out there*" an arrested terrorist is quoted in the same NYT issue as saying after the Bruxelles events.

Academia as well as government institutions have been aware quite a long time of this fact: terrorism thrives in some conditions, it may look like an "*impromptu*" individual act but requires instead a complex machinery to "*produce*" different kinds of terrors: shooting, kidnapping or bombing.

Security studies have time ago recognized the need to characterize the "modus operandi" of different groups and which conditions/events are to more likely to trigger attacks.

A main objective of TENSOR framework was to show how patterns of strategic behavior of different groups, identified analyzing large longitudinal data sets, can be linked to short term activity patterns identified analyzing feeds by "usual" surveillance technologies and that this fusion allows a better detection of terrorist threats. This point is debated in the thesis during explanations of TENSOR layered framework ability to leverage Long term reasoning and Short term surveillance into a more robust threat estimation.

There is now a general agreement that data driven pattern analysis, based on statistical learning, as TENSOR framework, will be the technology of choice for the development of new systems: still it has to be noted that terrorist offers some unique challenges: contrary to the wealth of data in digital marketing digital traces left by terror machinery are few and scattered, terrorism reinvents itself so it's difficult to match novel patterns into stored behavior so raising the alarm level is linked to an anomalous activity from a characteristic pattern.

1.1. Topic relevance

According to the definition in [Enders and Sandler, 2011] terrorism is the premeditated use or threat to use violence by individuals or sub-national groups to obtain a political or social objective through the intimidation of a large audience beyond that of the immediate victims.

Such a definition emphasizes that the true target of the anxiety-generating attacks is a wider public, who may pressure the government to concede to the demands of the terrorists.

Early studies mainly due to the absence of data, took a conceptual and historical approach to the study of terrorism focused on the definition of terrorism, the myriad causes of terrorism, the tactics of the terrorists, and the identity of the primary terrorist groups and movements (e.g. [Crenshaw, 1981]; [Wilkinson, 1986]).

Moreover, scholars of the analytical approach, such as William Landes [Landes, 1978], viewed terrorists as rational actors: If changes in terrorists' constraints, say through government policies, result in predictable behavioral responses by the terrorists, then terrorists are rational actors. Whether terrorists are rational, is widely debatable because they usually do not achieve their sought-after and stated objectives.

Indeed [Jones and Libicki, 2008] indicated 132 campaigns where the terrorist groups renounced terrorism and achieved their stated goals joining the political process.

The collection of event data gave a further boost to the analytical approach. In his landmark study of skyjackings, [Landes, 1978] used US Federal Aviation Association (FAA) data on skyjackings to estimate the deterrent effects of US antiterrorism policies against skyjackings during 1961–79.

After 2001 (9/11), there was an explosion of terrorism literature, both conceptual and analytical. More to the point for this thesis 9/11 opened a period of unprecedented investments by government and in turn companies into applications of advanced ICT into security and antiterrorism.

In addition to the proven importance of the matter in the literature, also from the economic point of view, terrorism has generated a significant commercial activities focus on providing different services, from physical protection against terrorist attacks (IBIS Corporation Counter-Terrorism International) [S1], to more widespread education on emergency management in case of attack (S2 Safety & Intelligence Institute) [S2], but also about the provision of a complete service for the anti-terrorist security as risk analysis, emergency response, industrial security program management, and security system technology implementation (Watermark Risk Management International, LLC) [S3].

Above references identify as antiterrorism became a major area of scientific and technological investigation spawning specific research domains and peculiar methods.

1.2. Contextualization and contributions

The domain addressed by this thesis is to investigate how data driven approaches might be relevant for modeling, detection and tracking of terrorist groups and their intents and provide actionable knowledge to LEA officers. In particular, this work is focused on development of automated techniques for detection and tracking of potential terrorist networks.

The starting point of the analysis concerns the study of the long term behavioral patterns of terrorist groups, that can be "observed" by considering past attacks. This information has been gathered from various research groups, who have created and shared several historical data sets.

One of the first repository used in the research community is the International Terrorism Attributes of Terrorist Events (ITERATE), a dataset that codes many variables (e.g. incident date, country location, target entity, attack type, casualties, perpetrators' nationalities, terrorist group, victims' nationalities, and logistical outcome) for transnational terrorist incidents. Currently, ITERATE's coverage is 1968–2011, with yearly updates. [Mickolus et al., 2012] ITERATE, like other event databases, relies on the news media. For hostage-taking incidents, ITERATE also contains negotiation variables, which have been invaluable in studies on hostage taking (e.g. [Sandler et al., 2009]). Another dataset is the RAND terrorist event database [RAND, 2012], which codes terrorist incidents for 1968–2009. After 1997, the RAND dataset began recording domestic terrorist incidents, so that it identifies transnational terrorist incidents for 1968–2009 and domestic terrorist incidents for 1998–2009.

The Minorities at Risk Organizational Behavior (MAROB) [S4] dataset is a subsidiary of the Minorities at Risk (MAR) Project. The project has identified 118 organizations representing the interests of all 22 ethno political groups in 16 countries of the Middle East and North Africa, operating between 1980 and 2004. While The Global Terrorism Database (GTD) records both domestic and transnational terrorist incidents [LaFree and Dugan, 2007]. For GTD, this partition of domestic and transnational terrorist incidents was first accomplished by [Enders et al., 2011] for 1970–2007 and has been updated by them through 2014.

In order to characterize the long term behavioral patterns of terrorist groups, in this thesis 2 databases have been analyzed: **MAROB** [S4] and **GTD** [GTD Codebook 2015]. These datasets include whole downloadable resources concerning codebook and data and they are focused on different aspects:

- MAROB is focused on organizations that represent the interests of ethnic groups around the world who have experienced state repression
- GTD represents a complete picture of terrorist incidents providing information about the actions carried out during an attack and the contest in which the attack occurred (contextualization).

A limitation of these data sets is that they provide just historical data and information about terrorist attacks. This information is useful for LEAs officers in order to detect and possibly prevent terrorist activities. However, in the last decade ICT technology was able to provide great help in this direction. In particular an ubiquitous surveillance of urban areas was provided through the use of a number of multi-purpose sensors in conjunction with emerging Big Data processing techniques for the data streams analysis [Sormani et al., 2016].

The long term behavioral patterns of terrorist and the data streams provided by sensors have been used in this thesis for two different tasks: online detection, that is focused on the early identification of a possible threat, and prediction, focused on the anticipation of potential future attacks.

The final result of this work is to develop a unique analysis and decisional ICT layered framework (TENSOR) for criticality assessment of terrorist related events at different time scales. In particular we contemplate:

- Long Term Reasoner (LTR) layer (section 6), based on an off-line clustering analysis of existing databases describing previous terrorist attacks
- Short Term Reasoner (STR) layer (section 5), based on Hidden Markov Models (HMM) to assess the criticality of a particular situation (Microenvironment).

LTR and STR are linked by a Medium Term Reasoner (MTR) layer (section 7) which leverages the output of STR into a decision, keeping into account the output of LTR.

Although research on automated information extraction from multimedia data has yielded significant progress, the development of automated tools for analyzing the results of information extraction and correlate them with specific events has been significantly slower. Moreover some features of terrorist networks, such as low Signal to Noise Ratio (SNR) (in the sense of sparse relevant observations superimposed upon a large background of benign ones) and a wide geographic distribution operating in different socio economic conditions, make them difficult to observe.

Terrorist networks are often a string of small cells, and interconnection among cells are purposely weak and therefore very difficult to detect. While a terrorist cell has its own "*modus operandi"*, in order to maintain a low profile, terrorist cells can move around geographically, alter their personnel and change their intended target.

A first relevant project aimed at developing an integrated layered framework for forecasting terroristic events is ASAM (Adaptive Safety Analysis and Monitoring system) [Popp et al., 2005] whose basic assumption is that terrorist networks can be evaluated using significant links between people, places and things that appear to be suspicious. To capture these relations and their changing nature ASAM uses dynamic Bayesian networks (DBNs) and Hidden Markov Models (HMMs). The HMMs detect the monitored terrorist activity and measure threat levels, whereas BNs combine the likelihoods from many different HMMs to evaluate the cumulative probability of terrorist activity.

Indeed the sequence of events might indicate, much more than the single events, a reason to be concerned: it may or may not arise from terrorist activity, but ought to be flagged for more careful scrutiny.

Our effort aims at increasing the resources available both to intelligence analysts, policy makers investigating reports of terrorist activities and LEA officers. With more effective tools for analysis, policy makers will have more and better information when planning some form of counter-terrorist action. The process of intelligence gathering and assimilation is largely automated but still unintegrated across the multiple agencies whose responsibilities might include responding to such events [Allanach et al., 2004].

Models based on relations and linked events are very efficient because instead of wasting resources interpreting the whole amount of data, they only have to look for significant links between the data. However, we want to point out that the number of instances of terrorism is (thankfully) very low and hence a "learning from data" approach is problematic.

The fundamental structure considered in this thesis is ascribable to HMMs. In our approach, a set of HMMs is used to model the evolution of threat level over time. In particular, for each monitored zone, a software component called *Micro-environment* is proposed for the online threat detection activity. Each Micro-environment interprets information provided by sensors (i.e. events detected in a specific zone) using a set of three HMMs. Each HMM implements a different interpretation strategy that takes in consideration the alert level of the monitored zone (i.e. the probability to have a threat in that zone) and the past sequence of events represented by the current HMM's state in order to associate a threat probability to every detected event. For every monitored zone, three different alert levels (Low, Medium and High) can be set – automatically or supervised by an expert. In particular the Low alert level refers to a normal condition (very low probability to have a threat), while at the opposite side, the High alert level refers to a very high threat risk condition.

ASAM system implements a hierarchical process, where lower levels correspond to HMMs, and higher levels are modeled through BNs [Singh et al., 2004]. Every HMM is used to represent a specific strategy to plan a type of terrorist attack (i.e. chemical attack, truck bomb attack). Each HMM's node represents a specific phase needed for the realization of a characteristic attack (e.g. members or money requirement, logistic organization, terroristic cells communications, etc.). Input for every HMM is the set of features that characterize each specific node (i.e. for communication are considered media activity, electronic communication activity, etc.). The BNs layer combines the updated information provided by different HMMs to evaluate the overall threat of inspected areas or social political events as probabilities distribution. Practically, BN represents the overarching terrorist schema and the HMMs (which are related to each BN node) represent more detailed terrorist operational methods.

Differently to ASAM framework, the TENSOR approach employs three different layers: at the lower level (STR), for every monitored zone, a Micro-environment (composed by a set of HMM) is used to model the evolution of threat level over time.

At the second level (MTR), we evaluate not only results from lower level as ASAM, but also zone-specific historical data, zone-specific alert level, spatial constrains between zones and predicted alert level provided by the highest level (LTR). MTR analysis is performed by a reasoning kernel that estimates the criticality of each zones (i.e. the alert levels). This module supports overall decision making by re-evaluating the alert level within a longer temporal scope than STR module and shorter temporal scope than LTR module.

As this thesis aims at producing timely proactive responses by relying on reliable forecasts about terrorists' actions, one can see a direct correspondence to event processing systems. CEP engines are designed for implementing logic in the form of queries or rules over continuous data flows. Typically, they include a high level declarative language for the logic definition with explicit support for temporal constructs. The need for CEP technology is rooted in various domains that require fast analysis of incoming information. Since the role of MTR reasoning module in the TENSOR approach is to process numerous incoming events of different nature and update the sensitivity of the STR in near real-time, we believe that event processing approaches fit requirements of the MTR module.

The last level, LTR, provides to MTR predictions of alert levels with respect to different types of physical zones (e.g. squares, churches, government buildings, etc.). This feature is not provide by ASAM framework. In order to carry out this activity the LTR uses information generated by STR (i.e. detected events and threats notifications) and external data stored in the terrorist data sources aforementioned. In particular external data sources are used to build a clustering based prediction model, that is used by LTR in its activities.

However, a further requirement was emerged during the design and development of TENSOR, and was related to the need of updating the STR's detection models. This activity needs a huge amount of information, which may be collected through the interactions between system (STR layer) and users, as presented in section 6.3.

The activity leading to this thesis has been partially conducted within the European project PROACTIVE (fp7 security call). Interactions with the partners have enabled the validation of TENSOR framework and the collection of users' feedback for further developments.

1.3. Thesis Outline

The thesis is organized as follow. In Section 2 we introduce the application domain addressed by this thesis. Section 3 presents an overview of the AI based event detection approaches related to security domain and the specific methods and algorithms investigated in this work. Section 4 presents the TENSOR framework that supply help at different users that work in counter-terrorism domain. Section 5, Section 6 and Section 7 present the three different reasoning components provided by the TENSOR framework (STR, LTR and MTR). Finally, in Section 8, conclusions are drawn.

2. Application domain

Security monitoring of urban areas is one of the main requests from citizens around the world. This was evidenced by the past series of terrorist attacks against New York's World Trade Center in 2001, Madrid' s train system in 2004, London Underground in 2007, Moscow Metro in 2010 and Paris in 2015. These incidents have exposed vulnerabilities of urban environments against terrorist actions, which mainly stem from their diversity, heterogeneity and complexity [Petris et al., 2014].

The challenges of the urban environment are unique with particular and very specific issues. The urban environment is characterized by the presence of buildings, city infrastructures and other man-made structures, over and underground, as well as considerable number of civilians [Hummel et al., 2012].

In order to cope with the complexity of operations in the urban environments, security and defense agencies have been increasingly turning in the last years to pervasive multi-sensory technologies for enhancing their ability to acquire, analyze and visualize events and situations. However, numerous challenges are associated with such technologies. The large scale nature of both the geographically dispersed environment and the volume of the data, the multiple distributed heterogeneous components that need to be assembled spanning sensors, sensor processing, signal processing components often from multiple vendors, are some of the issues that need to be addressed. Moreover, we need to take in consideration that some activities need to be automated (i.e. video analysis since manual observation of multiple camera is not possible, high-level intelligent reasoning for event inference), as these are features without which the added value of the system is limited.

Recent advances in multi-sensor systems and data analytics enable the development of systems that can collect and process information from a wide variety of sources, including structured and unstructured data, but also real-time and non-real time data. Closely related to multi-sensor systems is the internet-of-things (IoT) paradigm [Whitmore et al., 2015], which enables the orchestration and coordination of a large number of physical and virtual Internet-Connected-Objects towards human-centric services in a variety of sectors including logistics, trade, industry, smart cities and ambient assisted living.

A valuable solution could be a server-side middleware approach, running in a cloud computing environment. The middleware layer collects data through gateways or sink nodes. They have less control over sensor network operations. The components in this layer are unable to control low level operations such as routing though, they have more knowledge about the environment as they can analyze sensor data received through different sensors.

Todays' IoT systems rely on non-functional properties such as context awareness and semantic interoperability. Middleware systems can bundle those functionalities together to be reused in many applications [Wang et al., 2015]. These new type of middleware systems have more control of low level operations of network such as network routing, energy consumption, etc. This layer is much closer to the hardware but, it lacks the overall knowledge about the environment.

It is important that systems providing advanced capabilities for urban security and surveillance advance and integrate all above functionalities in order to handle multiple heterogeneous sensors suitable for the monitoring activity of urban environment. At the same time, these systems need to support JDL fusion level mandates by deploying various fusion techniques and algorithms [Hummel et al., 2012].

Modern multi-sensor and IoT systems, have been extensively used in order to collect and visualize information about the surrounding environment, based on sensor information fusion and common operational picture generation tools. However, despite their suitability, they have not been used for predicting terroristic attacks. In this thesis we introduce a layered framework, integrated in an IoT system for the prediction of terrorist attacks in urban environment, that provides help at the different users of the system. The framework process information sources at different abstraction levels (e.g. sensor information, police patrol inputs, external semantic crafted data sources) and methods designed to act (i.e. simulate) various expert user roles indicated as crucial in the intelligence analyst work flow (i.e. operational, tactical and strategic user roles).

2.1. Users classification

In order to design our layer framework we taking into account both the abstraction levels of the potential information sources (sensor information, police patrol inputs, news events, external semantic crafted data sources) and the expert user roles that are currently defined as crucial in the intelligence analyst flow for analyzing/detecting potential terrorist threats.

According to the classification proposed by The US Department of Defense in [Department of Defense, 2010], three different type of users were identified: operational, tactical and strategic users. These users operate at three different levels as follows:

- The Operational level refers to regional Intelligence services with more focus on the regional threat level. In this level local terrorist actions may increase the alert state at the regional level but still may not affect the National level.
- The Tactical level refers to local Intelligence Units (e.g., intelligence units of local police). At this level the alert state changes with even small scale terror actions. The threat assessment is limited in time and space.
- The Strategic level is a National Intelligence Service, where threat assessment covers the whole country. At the National level the alert state does not change instantly based on isolated small scale terrorist events. The threat assessment is broader in space and time.

It is important to note that, serious terror actions with political consequences at a local level may have serious impact to the National threat level based again on the “gravity” of the event. The level of gravity cannot be precisely defined and it is up to National Authorities and the higher political level to define the seriousness of a terror event based on National policies.

In a counter-terrorism applications, *operational* users have the aim to identify and notify tactical users about suspicious situations in a physical environment of limited size and complexity (i.e., a city zone) taking into account short histories of events provided by sensors (human or device).

Tactical users need to analyze a medium term history (sensor data and threats notification) coming from different zones in order to infer the sensitivity/criticality (i.e. the alert levels, see section 2.2) of the current situation in each monitored zone. Finally, *strategic* users work in order to predict the criticality of each monitored zone tacking into account historical data and external data sources.

2.2. Alert level

An important outcome of an end user workshop was a specific requirement that has been considered for the framework design. In this workshop with domain experts from EU countries, users have expressed the need to have the possibility to summarized in a synthetic way the alert level (i.e. the criticality level) of each monitored zone. This user requirement can be paragoned to the UK Threat Level [UK Mi5] and the DEFCOM [Department of Defense, 2010] level used by the United States Armed Forces. These different alert levels give a broad indication of the likelihood of a terrorist attack and a specific reaction policy that security agencies have to adopt (Table 1).

Table 1 UK and US Threat levels.

UK Threat Levels		**US DEFCOM Levels**	
Critical	An attack is expected imminently.	**DEFCON 1**	Nuclear war is imminent
Severe	An attack is highly likely.	**DEFCON 4**	Next step to nuclear war
Substantial	An attack is a strong possibility.	**DEFCON 3**	Increase in force readiness above that required for normal readiness
Moderate	An attack is possible, but not likely.	**DEFCON 2**	Increased intelligence watch and strengthened security measures
Low	An attack is unlikely.	**DEFCON 1**	Lowest state of readiness

Following ideas for the first three levels of the UK Threat Levels and the domain expert workshop, in this thesis a simpler ranked alert levels are introduced. In particular, a new set of level's labels (Low, Medium and High) was adopted in order to simplify the association between the alert classes and the likelihood of a terrorist attack.

2.3. Threat type overview

In this section we focus on the various types of threats that could occur. Note that threats could be malicious threats due to terror attacks or non-malicious threats due to inadvertent errors. According to [Mahmood and Rohail, 2012] the types of terrorist threats include:

- Information related terrorism
- Non-information related terrorism
- Bio-terrorism and chemical/nuclear attacks

However all the mentioned threats type can be grouped into two categories; *non real-time threats* or *real-time threats*. In a way all threats are real-time (a non real-time threat could turn into a real-time threat) as we have to act in real-time once the threats have occurred. However, some threats are analyzed over a period of time while some others have to be handled immediately. Biological, chemical and nuclear threats have to be handled in real-time, other threats instead do not have to be handled in real-time, for example consider the behavior of suspicious people that maybe they are member of terrorist organization. In this case, one has to monitor these people, analyze their behavior and predict their actions.

2.3.1. **Non-information related terrorism**

Non-information related terrorism means terrorism not focus on computers and networks. In this type of threat are present terrorist attacks like car bombing, vandalism, bomb explosion, building intrusion and so on. These threats can be perform from people inside an organization that attack others around them (e.g. an agent from an intelligence agency committing espionage). This type of threats is called *insider threat*, while if the threat is carry out by an external person (e.g. a mentally deranged person of a society, a member of a foreign terrorist group) are called *external threats*.

As an example, transportation systems security violations can be an example of external (or internal) threat. Buses, trains and airplanes are vehicles that can carry tens of hundreds of people at the same time and any security violation could cause serious damage and even deaths.

2.3.2. **Information related terrorism**

Information related terrorism refers to attacks that damage electronic information (computers and/or networks components). Examples of these attacks can be viruses or information security violations. Note that threats attacks can occur from outside or from the inside of an organization.

Cyber-terrorism is one of the major terrorist activity, there are so much information available electronically that an attack on a network (computers, databases) could be devastating to businesses. These intruders may be human intruders or Trojan horses set up by humans. Intrusions can also occur on databases. Intruders posing as legitimate users can pose queries such as SQL queries and access the data that they are not authorized to know. Information security violations typically occur due to access control violations.

There are numerous security attacks that can occur from the web and these attacks are applicable to any information system such as networks, databases and operating systems. These threats include access control violations, integrity violations, sabotage, fraud and infrastructure attacks.

Other examples of these threats are credit card frauds, but a more serious theft is identity theft in which one assumes the identity of another person and start to carried out all the transactions under the other person’s name.

2.4. A general architecture used in urban security domain

In urban security monitoring the role of IoT systems consist, mainly, in identification, management and potential prediction of characteristic events that reveal the likelihood of a terrorist attack. While traditional ambient security systems were focused on the extensive use of arrays of single-type sensors [Monekosso and Remagnino 2007], modern systems aim to combine information coming from different types of sources [Petris et al., 2014].

Data fusion is the process of combing information from a number of different sources to provide a robust and complete description of an environment or process of interest [Azimirad and Haddadnia, 2015]. Data fusion is of special significance in any application where a large amount of data must be combined, fused and distilled to obtain information of appropriate quality and integrity on which decisions can be made.

A standard model for data fusion that was proposed by the US Department of Defense to facilitate discussion, component reuse and system integration is the Joint Directors of Laboratories (JDL) data fusion model. This model offers a multi-level functional model that describes how processing is organized in a military data fusion system and more generally, the JDL data fusion model is recognized as a de facto standard in data fusion issues [Foresti et. al., 2015].

2.4.1. **JDL model**

The JDL Model [Hall and Llinas, 2001] [Klein, 2004] is a well-established reference model, which provides a sound basis for the identification of the major abstraction levels to be considered in the proposed solution approach. In this section we summarize the layered JDL model. Figure 1 recalls the functionalities of the JDL levels and the information flows between levels, no matter how information is processed inside each level and how processing activities are grouped in software components.

Figure 1 highlights that levels 0 – 3 are directly involved in the real-time upstream information and processing flow from data sources to end users. JDL level 4, whose functionalities belong to medium and long term activities, is not directly involved in the real-time upstream information flows from sensing devices to end users.

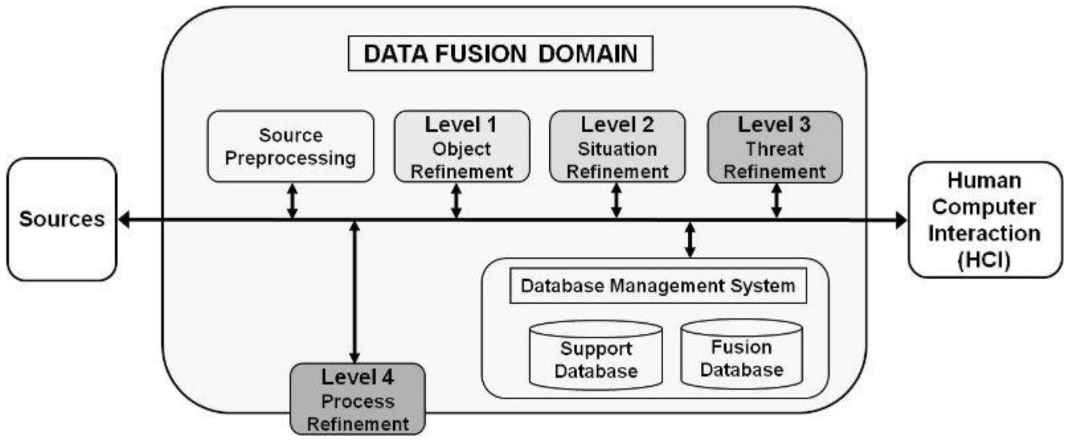
DATA FUSION DOMAIN
Source Preprocessing
Level 1 Object Refinement
Level 2 Situation Refinement
Level 3 Threat Refinement
Sources
Human Computer Interaction (HCI)
Database Management System
Support Database
Fusion Database
Level 4 Process Refinement

Figure 1 JDL Layers.

JDL Level 0

Level 0 (Source pre-processing in JDL parlance) is in charge of processing raw data from individual sensors. Its activities include filtering and extraction of low-level features (e.g., barycenter, area and speed of a blob in the reference system of the sensor image), which model low-level perceptions [Sabzevar, 2015]. A perception is characterized by:

- Timestamp (though not necessarily, in a global reference time);
- Perception type, dependent on the sensor type (e.g., image blob or acoustic perception);
- Features dependent on the sensor type.

Level 0 activities deal with strictly local data from individual sensors. Perceptions are located in a sensor-specific space (e.g., the image plane of a camera). Signal refinement is bound to strong real-time constraints (tenth of ms) and has a high frequency (tenths of events for second for each sensor).

JDL Level 1

Level 1 (Object refinement) is in charge of translating perceptions into states of objects that model significant domain entities (e.g., persons and cars) in a local context [Golestan et al., 2016]. Possibly this can be done by fusing information from a set of related sensors. Object states are characterized by:

- Unique id of the object in a local context;
- Object type;
- Timestamp in a global reference time;
- Localization in a local reference system (e.g., the area covered by a sensor or by a set of sensors);
- Sensor preprocessed features (e.g., the speed of a car).

Level 1 activities are local, as they deal with information from individual sensors or from sets of strongly related sensors (e.g., a camera and an associated microphone). Objects are localized in a local but sensor-independent space (e.g., the Cartesian co-ordinates in the area covered by a sensor). The generation of object states is bound to strong real-time constraints, possibly an order of magnitude lower than event generation (say hundreds of ms). The same holds for the frequency (say from one to ten states for second).

JDL Level 2

Level 2 (Situation refinement) is in charge of fusing states of objects from Level 1, which are defined in sensor-local contexts, into higher level situations in the context of a micro-scenario [Pires et al., 2016]. A micro-scenario models a target (e.g. a building, a metro station, a limited set of streets, etc.), which can be monitored by a set of information providers (i.e. devices and/or persons). Each micro-scenario defines a spatial reference system (e.g. the map of the building).

A situation is the abstract representation of a set of observations of one or more objects, deriving from the temporal and positional fusion of the states of objects coming from Level 1 (for example, observations of a car from different cameras around a building is fused into a situations modelling the movement of the car in the reference space of the micro-scenario). Situations are characterized by:

- Unique id of the situation (and of the objects they include) in the spatio-temporal context of the micro-scenario;
- Situation type;
- Timestamp in a global reference time;
- Localization in the spatial reference system of the micro-scenario (e.g., the GPS reference system or a route model in an urban area);
- High-level symbolic features (e.g., the movement of a group of logically related people, though they are not physically close).

Level 2 activities are local to individual micro-scenarios, as they fuse and contextualize information items that, tough coming from different sources, are related to a specific micro-scenario. The generation of situations is bound to real-time constraints that depend on end-user requirements (say one second) and on the requirements of Level 3.

JDL Level 3

Level 3 (Threat refinement) is in charge of interpreting situations (in this work, threat detection) by assigning them threat levels which are presented to the final users (typically, through GUIs) [Sabzevar, 2015]. Threats are characterized by:

- Unique id of the threat (and of the situations it includes) in the micro-scenario context;
- Threat type and level;
- Timestamp in a global reference time;
- Localization in the spatial reference system of the micro-scenario (e.g., the GPS reference system or a route model in an urban area);
- High-level symbolic features.

Level 3 activities are bound to real-time constraints, which are less strong than those of lower levels (say a few seconds: we deal with terrorism in a urban environment, not with driving a fighter). Moreover, the identification and notification of threats falls inside a “man-in-the-loop” scheme, so that the timing constraints are consistent the decision delays of the users. Even the frequency is relatively low if compared with the frequency of lower level activities and flows, at least to avoid information overflow.

JDL Level 4

Level 4 (Process refinement) [Pires et al., 2016] is a meta-layer, whose major role is to generate, train and tune the fusion models, which are exploited by lower layers of the system and, in particular, by the Level 3 (threat detection). Level 4 is not directly involved in the real-time upstream information flow from sensing devices to end users. L4 activities are not bound to strong real-time constraints.

2.5. The data available in security domain

The majority of urban surveillance systems focus on situation awareness and common operation picture generation, and pay limited or even no attention to the task of predicting potential terrorist actions. This is a significant limitation given that the anticipation of terrorist actions could allow Law Enforcement Agencies (LEAs) to proactively deal with them thereby minimizing their adverse effects. The development of systems for predicting potential terrorist attacks hinges on devising appropriate reasoning and analytics techniques that could operate over information collected from various sources, including sensor sources, human operators and open source information [Sormani et al., 2016].

These techniques can be typically classified as Big Data processing schemes [Wu et al., 2014], since they have to deal with large data volumes stemming from multiple heterogeneous sources and featuring varying velocities. A serious set-back to the development and operation of appropriate reasoning and analytics techniques for anticipating terrorist events is the lack of ground-truth datasets, which could be used to train the respective reasoners. Typically, LEAs do not possess large sequences of events that are associated with terrorist attacks, since their databases are usually limited to the set of events that have been observed before the occurrence of past attacks.

In this subsection, after a brief excursus on the available datasets containing different information about terrorist activities, some tools and techniques for simulating terrorist attacks as a means of producing training data sets are presented.

2.5.1. Security databases

The datasets available on the web are focused on various terroristic aspects, starting from the description of the terrorist groups (e.g. MAROB dataset) to the description of terrorist attacks with targets and weapons used (e.g. GTD dataset) or more general datasets containing news about terroristic threats (e.g. Rand dataset). We focused our attention on the analysis of the following datasets:

- **Iterate** (International Terrorism –Attributes of Terrorist Events)
- **GTD** (Global Terrorism Database)
- **Rand** (Rand Database of Worldwide Terrorism Incidents)
- **MAROB** (Minorities at risk Organizational Behaviour)
- **ACLED** (Armed Conflict Location and Event Dataset)
- **WITS** (Worldwide Incidents Tracking System)

Almost all of these datasets are accompanied with a codebook that describes the information contained (Table 2).

Table 2 Short description of terrorist datasets.

	ITERATE	RAND	GTD	MAROB	ACLED
Overview	International Terrorism–Attributes of Terrorist Events	Rand Database of Worldwide Terrorism Incidents	Global Terrorism Database	Minorities at Risk Organizational Behavior	Armed Conflict Location and Event Dataset
DB Operator	University Michigan and Vinyard Software	RAND (Original PIs: Jenkins, Hoffman)	START University Maryland (LaFree)	Department of Homeland Security	International Peace Research Institute (PRIO)
Website	www.icpsr.umich.edu/icpsrweb/ICPSR/studies/7947?q=ITERATE	www.rand.org/nsrd/projects/terrorism-incidents	www.start.umd.edu/gtd	www.cidcm.umd.edu/mar/data.asp	www.acleddata.com/
Unit of analysis Scope of events	Events International	Events International + domestic	Events International + domestic	Events International	Events International + domestic
Time span	1968- 2008	1968- 2009	1970- 2011	1980 and 2004	1997- 2012
# of events	~13,000 ~36,000	~13,000 ~36,000	~104.700	~1790	~75,000
# of features	42	15 + narrative description	~124	~175	25
Coding transparency and consistency	Provides detailed codebook	Provides basic information for coding of variables on website	Provides criteria for incident inclusion and coding scheme in a codebook.	Provides detailed codebook	Provides detailed codebook
Online availability/	Partial/incomplete	Full	Full	Full	Full
Free Download	No	Partially/on request	Yes	Yes	Partially/on request
Reference	Mickolus, Edward F. InTErnational terroRism: Attributes of Terrorist Events, 1968-1977 [ITERATE 2]. ICPSR07947-v1. Ann Arbor, MI: Inter-university Consortium for Political and Social Research, 1982.	"RAND Database of Worldwide Terrorism Incidents"http://smapp.rand.org/rwtid	National Consortium for the Study of Terrorism and Responses to Terrorism (START). (2012).	Asal, Victor, Amy Pate and Jonathan Wilkenfeld. 2008. Minorities at Risk Organizational Behavior Data and Codebook Version 9/2008	Raleigh, Clionadh, Andrew Linke, Håvard Hegre and Joakim Karlsen. 2010. Introducing ACLED-Armed Conflict Location and Event Data. Journal of Peace Research 47(5) 1-10.

According to the specific reported in Table 2, we focus only on two fully available datasets, which include whole downloadable resources concerning codebook and the full data (all instances and features).

2.5.2. **Minorities at Risk Organizational Behaviour (MAROB)**

The first analyzed data source is the MAROB dataset (Minorities at Risk Organizational Behaviour) dataset, a subsidiary of the Minorities at Risk (MAR) Project. The purpose of this project is to answer fundamental questions focusing on the identification of those factors that motivate some members of ethnic minorities to become radicalized, to form activist organizations, and to move from conventional means of politics and protest into violence and terrorism. Focusing initially on the Middle East and North Africa, the MAROB project provides information on the characteristics of those ethno political organizations most likely to employ violence and terrorism in the pursuit of their perceived grievances with local, national, or international authority structures. MAROB is directed by Jonathan Wilkenfeld (University of Maryland), Victor Asal (University at Albany), and Amy Pate (University of Maryland).

The project has identified 118 organizations representing the interests of all 22 ethno political groups in 16 countries of the Middle East and North Africa, operating between 1980 and 2004.

The project developed a set of criteria for the inclusion of organizations into the MAROB dataset, specifically satisfying at least one of these:

- The organization makes explicit claims to represent the interests of one or more ethnic groups and/or the organization’s members are primarily members of a specific ethnic minority.
- The organization is political in its goals and activities.
- The organization is active at a regional and/or national level.
- The organization was not created by a government.
- The organization is active for at least three consecutive years between 1980 and 2006.

Organizations were selected on the basis of their basic longevity. This was operationalized in the following manner: the first year that an organization is mentioned in a source as being active, it is put on a “watch list” for potential inclusion. Once the organization is mentioned in sources for three consecutive years, it is included in the dataset, coded from the first year of the three consecutive years. If an organization included in the dataset disappears from source material for five consecutive years, it is no longer coded for following years. If after that time, it is again mentioned for three consecutive years, it is again included but as a separate organization.

The dataset contains information on 1789 terrorist attacks occurred from 1980 to 2004 focusing on the description of the organization behind those attacks operating in 16 countries, as depicted in Figure 2.

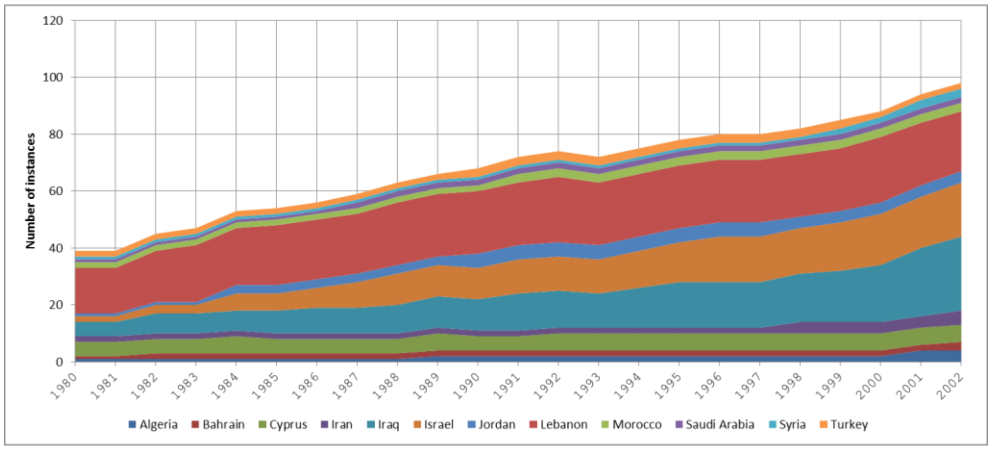

Figure 2 MAROB attacks for year grouped by country.

2.5.3. **Global Terrorist Database**

As reported in Table 2 one of the data source with the highest number of recorded events is the Global Terrorism Database (GTD) [GTD Codebook 2015], an open-source database including information on terrorist events around the world from 1970 through 2014. Unlike many other event databases, the GTD includes systematic data on domestic as well as transnational and international terrorist incidents that have occurred during this time period and includes 141.996 instances. For each GTD incident, information is available on the date and location of the incident, the weapons used and nature of the target, the number of casualties, the group or individual responsible and other information (each instance is encoded by 124 features).

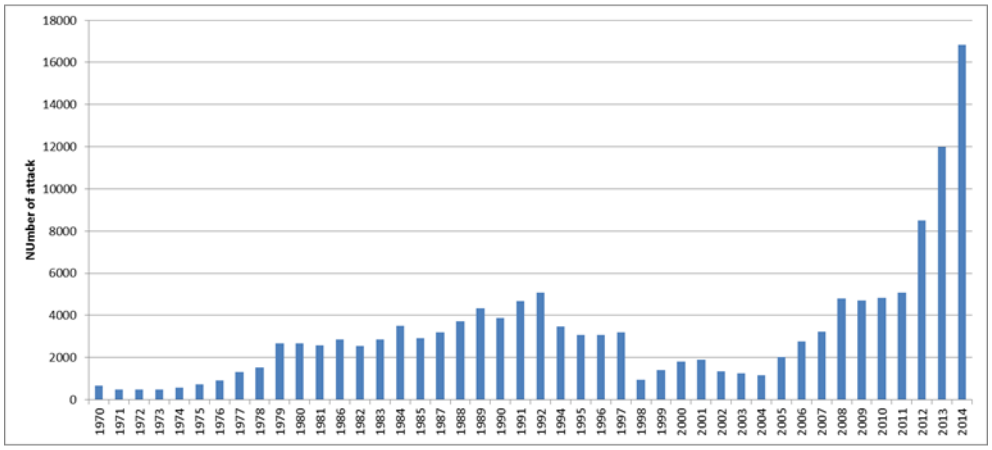

Figure 3 Trend attacks in the period 1970- 2014.

The Global Terrorism Database was created in 2001 when researchers at the University of Maryland obtained a large database originally collected by the Pinkerton Global Intelligence Services (PGIS). From 1970 to 1997, PGIS trained researchers to identify and record terrorism incidents from wire services, government reports, and major international newspapers in order to assess the risk of terrorism for their clients. With funding from the National Institute of Justice, the Maryland team finished digitizing the original Pinkerton data in December 2005, making corrections and adding additional information wherever possible. PGIS lost data for 1993 in an office move and these data have never been fully recovered.

As stated in the introduction, this database represents a validated source of information about historical terroristic attacks for the LTR prediction analysis. In this dataset is possible to distinguish between two principal categories of features:

- The “context” features describing the context of the attack, (e.g. The type of the target)
- The “action” features describing of the actions performed during the attack (e.g. Type of action performed)

In order to introduce the link between the detected events (provided by sensors) and GTD events, we would like to focalize the attention on the analysis of the following features: “target type” and “attack type”.

Target type

This feature can assume 22 different values representing the different type of target of each terroristic attack (ex. Business, government, etc.). In order to have an overview of the values assumed by this feature in the database, we propose two different figures depicting the distribution of the different attacks grouped by year and type.

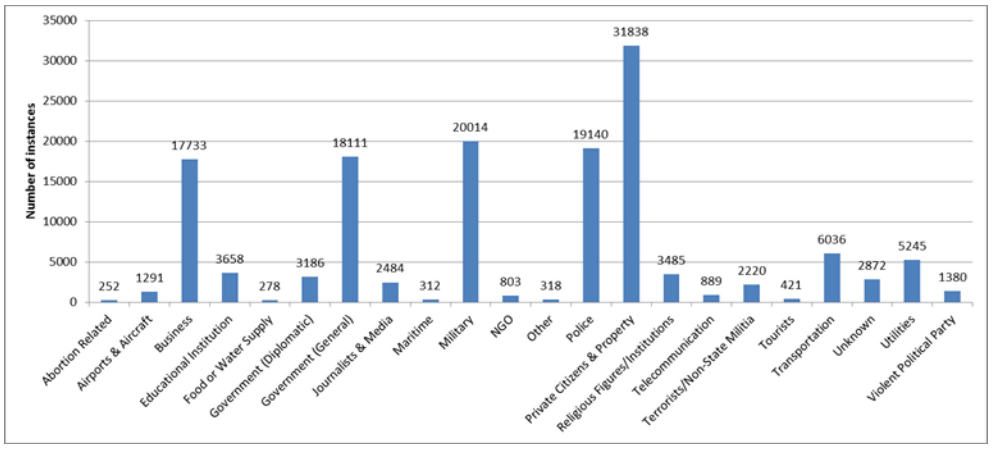

Figure 4 Number of instances for each target type.

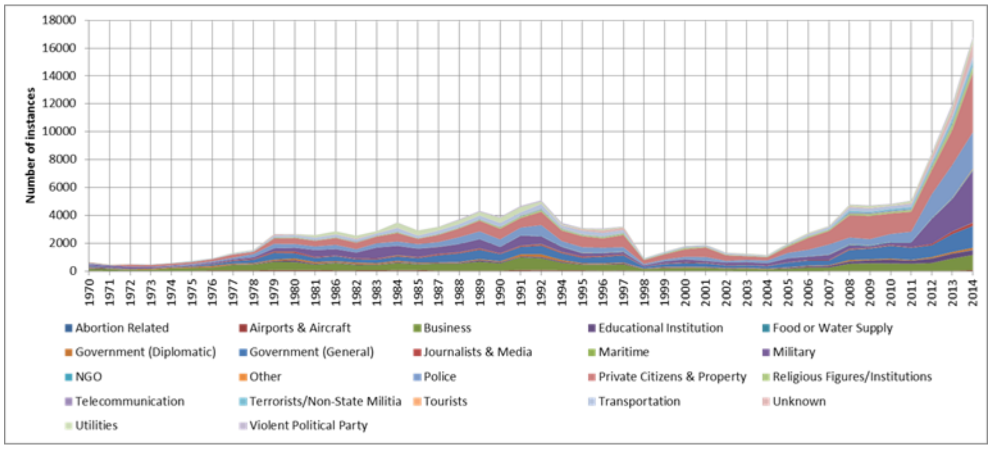

Figure 5 Attacks per Year grouped by target type.

For a better understanding of each kind of target depicted into the previous figures, we report a brief explanation of each possible value assumed by the variable "target type". This description has been extracted from the GTD codebook [GTD Codebook 2015].

- **Businesses** are defined as individuals or organizations engaged in commercial or mercantile activity as a means of livelihood
- **Government** Any attack on a government building; government member, former members, including members of political parties in official capacities, their convoys, or events sponsored by political parties.
- **Police** This type includes attacks on members of the police force or police installations; this includes police boxes, patrols headquarters, academies, cars, checkpoints, etc.
- **Military** Includes attacks against army units, patrols, barracks, and convoys, jeeps, etc.
- **Abortion related** Attacks on abortion clinics, employees, patrons, or security personnel stationed at clinics.
- **Airports & Airlines** An attack that was carried out either against an airplane or against an airport. Attacks against airline employees while on board are also included in this value..
- **Government** Attacks carried out against foreign missions, including embassies, consulates, etc.
- **Educational institution** Attacks against schools, teachers, or guards protecting school sites. Includes attacks against university professors, teaching staff and school buses.
- **Food or water supply** Attacks on food or water supplies or reserves are included in this type. This generally includes attacks aimed at the infrastructure related to food and water for human consumption.
- **Journal & media** Includes, attacks on reporters, news assistants, photographers, publishers, as well as attacks on media headquarters and offices.
- **Maritime** (includes ports and maritime facilities) Implies civilian maritime. Includes attacks against fishing ships, oil tankers, ferries, yachts, etc.

- **Non-governmental organizations** Includes attacks on offices and employees of non-governmental organizations (NGOs). This type include large multinational non-governmental organizations such as the Red Cross and Doctors without Borders.
- **Other** This type includes acts of terrorism committed against targets which do not fit into other categories.
- **Private citizens & Property** This type includes attacks on individuals, the public in general or attacks in public areas including markets, commercial streets, busy intersections and pedestrian malls.
- **Religious figures/institutions** This type includes attacks on religious leaders, (Imams, priests, bishops, etc.), religious institutions (mosques, churches), religious places or objects (shrines, relics, etc.). This type also includes attacks on organizations that are affiliated with religious entities that are not NGOs, businesses or schools
- **Telecommunication** This includes attacks on facilities and infrastructure for the transmission of information.
- **Terrorist/Non-state militias** Terrorists or members of identified terrorist groups within the GTD are included in this value. Membership is broadly defined and includes informants for terrorist groups, but excludes former or surrendered terrorists.
- **Tourists** This type includes the targeting of tour buses, tourists, or "tours." Tourists are persons who travel primarily for the purposes of leisure or amusement. Government tourist offices are included in this value.
- **Transportation** Attacks on public transportation systems are included in this type. This can include efforts to assault public buses, minibuses, trains, metro/subways, highways (if the highway itself is the target of the attack), bridges, roads, etc.
- **Unknown** The target type cannot be determined from the available information.
- **Utilities** This type pertains to facilities for the transmission or generation of energy. For example, power lines, oil pipelines, electrical transformers, high tension lines, gas and electric substations, are all included in this type.
- **Violent political parties** This type pertains to entities that are both political parties (and thus, coded as "government" in this coding scheme) and terrorists.

In order to complete the information presented into the previous section, a brief description of the main types of action performed by terrorists during the execution of an attack is reported. In GTD this feature is called "type of action" and can assume 9 different values as depicted in the Figure 6.

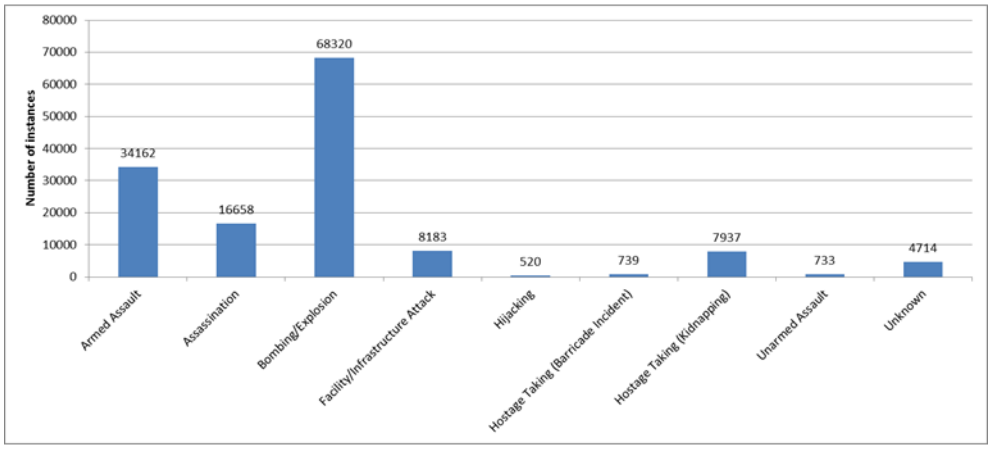

Figure 6 Number of instances for each action performed.

As for the target nature analysis, we report in the number of attacks grouped by year for each type of action performed by terrorists.

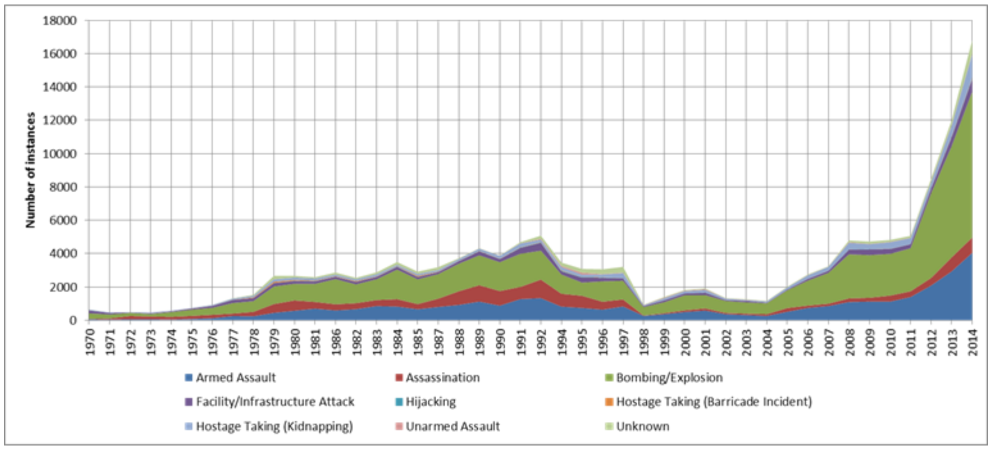

Figure 7 Attacks for year grouped by actions performed.

Also for this feature, a brief summary of the description of each possible value assumed by the variable "type of action performed" is reported [GTD Codebook 2015].

- **Assassination** An act whose primary objective is to kill one or more specific, prominent individuals.
- **Armed assault** An attack whose primary objective is to cause physical harm or death directly to human beings by use of a firearm, incendiary, or sharp instrument (knife, etc.).
- **Bombing/Explosion** An attack where the primary effects are caused by an energetically unstable material undergoing rapid decomposition and releasing a pressure wave that causes physical damage to the surrounding environment.

- **Hijacking** An act whose primary objective is to take control of a vehicle such as an aircraft, boat, bus, etc. for the purpose of diverting it to an un-programmed destination, force the release of prisoners, or some other political objective.
- **Hostage taking** (barricade incident) An act whose primary objective is to take control of hostages for the purpose of achieving a political objective through concessions or through disruption of normal operations.
- **Hostage taking** (kidnapping) An act whose primary objective is to take control of hostages for the purpose of achieving a political objective through concessions or through disruption of normal operations. Kidnappings are distinguished from Barricade Incidents (above) in that they involve moving and holding the hostages in another location.
- **Facility/infrastructure attack** An act, excluding the use of an explosive, whose primary objective is to cause damage to a non-human target, such as a building, monument, train, pipeline, etc.
- **Unarmed assault** An attack whose primary objective is to cause physical harm or death directly to human beings by any means other than explosive, firearm, incendiary, or sharp instrument (knife, etc.).
- **Unknown** The attack type cannot be determined from the available information.

As already clear in the previous sections, during the reported analysis of the two data sources (GTD and MAROB), we mapped the features reported in both dataset to those that are required for the alert prediction activity. While MAROB dataset is focused on organizations that represent the interests of ethnic groups around the world who have experienced state repression, the GTD database represents a complete picture of terrorist incidents. In fact, GTD provides more information about the characteristics of an attack (e.g the instruments used: guns, bombs) and about the contextualization of that. This contextualization is strongly related to the description of the physical monitored environments used in this thesis. Also the number of events stored into GTD is bigger than those of MAROB, as reported into Table 2.

For all these reasons GTD is selected as data source for train and test the long term clustering prediction approach (see section 6.1 and 6.2).

2.5.4. **Gaming data**

Computerized serious games have received a lot of attention lately as they have been used for a variety of educational and training purposes, to raise awareness and funds for good causes, to collectively solve difficult and important problems, to detect, evaluate and enhance social and business skills, as well as for diagnostic and therapeutic purposes [Susi et al., 2007].

These games are typically built based on realistic requirements provided by LEAs. Therefore, they are appropriate not only for training officers in effectively engaging in terrorist situations, but also for:

- Developing expert system rules that can be used to raise alarms on the risk of terrorist attacks;

- For generating sequences of data that will be used to train expert systems (such as reasoners) into classifying situations and reaching automated decisions about the probability of an impending attack.

Role playing games have a long history of use in armed and security forces training even before computerized games emerged [Smith, 2009]. Computerized games in counter-terrorism activities have also been developed and used to train security forces and first responders. Most of them fall into two major categories: active engagement combat like missions against terrorists attacking a target [Silverman et al., 2006] or crisis management during and after terrorist attacks. In the first category there exist even some very popular commercial computer games such as Counter Strike. Finally, there are a few games aiming to train security forces at screening suspects, setting up checkpoints, patrolling land and sea borders and inspecting suspicious vehicles or vessels [Caspian learning, 2008].

There are few games aiming to train the security forces to efficiently deploy resources and predict terrorist attacks in an urban environment. A prominent such game is SIBILLA [Bruzzone et al., 2009]. The main focus in SIBILLA is in the collaboration between different agencies by sharing information in a way which maximizes their chances not only to prevent terrorist attacks but to beat other agencies in doing so. In SIBILLA each player assumes the role of a collaborating agency, while the role of the terrorists is carried out by the computer program. Another simulator training tool that can be used to train camera operators to spot suspicious behavior (either terrorist or criminal) in camera feeds from crowded areas is EyeObserve [VSTEP]. In terms of technology, several serious games and simulators employ ergonomic and motivating technologies for engaging end-users, such as virtual reality technologies [Zyda, 2005].

Most of the above-listed games are used for training and simulation, but are not used for generating data sets that can be used for prediction of real-life settings.

A lot of attention has been also given into modeling terrorist action and counter-action using game theoretic analysis tools. Since the relevant literature is quite large to be summarized in a couple of paragraphs, we refer the reader to the overview papers [Sandler and Siqueira, 2009] and [Eiselt et al., 2013] and references within. [Sandler and Siqueira, 2009] provides an overview of game theoretic modeling of various aspects of the anti-terrorist war. In particular, it investigates the government's allocation of a fixed budget to counter attacks against potential targets, the choice between proactive and defensive countermeasures, along with the impact that domestic politics has on this choice, the interaction between political and militant factions within terrorist groups, the role of asymmetric information, and suicide terrorism. [Eiselt et al., 2013] uses game theory models to analyze three important counter-terrorism tasks: the detection and neutralization of terrorist cells, the fortification and protection of assets and the optimal evacuation of people from an area hit by a terrorist attack.

There are works which combine game theory and serious games. In the scope of a prominent example [Silverman et al., 2006], the authors use a game theory matrix of utilities in order to build "rational actor" models for decision making that are used in implementing software agents representing terrorists in simulated training environments.

3. State of the art

The relevance of Artificial Intelligence (AI) in event detection has been growing steadily bringing many approaches to the attention of researches and practitioners, whose mentions is really outside the scope of this work. For a general overview see [Atefeh and Khreich, 2015] and [Zuech et al., 2015]. More to the point of this thesis we analyze in more details AI based event detection related to security (section 3.1) and modelling algorithmic approaches that have been specifically investigated (section 3.2).

3.1. AI threat detection approaches

In order to detect threats, data mining is usually applied on data that has been gathered over extended period of time, with the goal to analyze data and make deductions and predict future trends. Ideally it is used as a decision support tool. In the literature many algorithms have been investigated for threat and/or anomalous situation detection, in this section an overview of this methods is resented.

3.1.1. Information Extraction

A variety of information extraction techniques can be applied in analysis of terrorists and extremism, including topic mining and summarization. Websites and forum frequented by these groups are particularly rich sources of information. The main body of research tends to focus on either white supremacist groups in the United States or Islamic fundamentalists.

[Yang and Ng, 2008] relate how a clustering opinion-extraction method targeted specifically at opinions expressed in online discussion is experimented on a corpus drawn from MySpace.

[Jayanthi and Sasikala, 2011] describe a process of mining hyperlinks from terrorist webpages as part of link analysis, but only a simulation of output from a toolkit is provided.

[Inyaem et al., 2009] focus on gathering open-source information about terrorism via summarization of web-based news articles. Though some details appear obscured by poor translation, the authors seem to find support for an ontological approach to detection of terrorist events, comparing this approach to a gazetteer and a grammatical parser in an evaluation on Thai news articles.

3.1.2. Machine Learning

Machine learning is deployed both in identifying terrorists from their online footprint and in detecting terrorism-related activities.

Identify terrorists from online footprint

[Cheong and Lee, 2011] explores microblogging within the terrorism informatics domain. They perform an observational analysis of the Twitter network's response to two real-life terrorist events, and use this as inspiration for the design of an information gathering framework. They later apply the framework to a synthetic dataset of events which share some properties with terrorism events. They also apply a variety of common machine-learning analyses to their dataset in an exploratory manner.

[Tinguriya and Kumar, 2010] suggest a self-organizing map (SOM) approach to classifying web users from usage data. They provide no evaluation or appraisal of their proposed system, and minimal description of its proposed operation.

In [Endy et al., 2010] the authors have developed what they term an intelligent search procedure for web mining cyberterrorism information, feeding a vector representation of 600 articles, half related to cyberterrorism, into an SOM, the results of which they then briefly dissect. Their presentation of the SOM as a heat-colored grid seems ill-suited for law enforcement analysts.

[Sahito et al., 2011] links streams of Twitter data to other resources through open data mechanisms. They apply named entity recognition to the content of Tweets. They mention the terrorism domain, their aim being to allow for structural links to entities to be imposed on unstructured Twitter data to better allow law enforcement to parse and respond to events detected via Twitter. However, the implementation of this is relegated to future work.

[Shen and Boongoen, 2012] uses a qualitative formalism as the basis for a fuzzy analysis, applying this to link analysis and the determination of aliases. They evaluate their system against unspecialized, unsupervised learning systems on a constructed terrorism dataset gathered from web articles, an author publication dataset (DBLP) and an email dataset. Their system appears to outperform a number of similar link based algorithms.

[Yang et al., 2012] focus on identifying extremist content in social media sites, drawing their design inspiration from biological immune systems. They build a mathematical representation of lymphocytes which incorporates lexical, sentiment, and syntactic features of text as a precursor to a semi supervised classification system. In an evaluation of this system on violent messages scraped from a white supremacist web forum, their system outperformed two benchmark labelling systems.

[Chung, 2012] studies appropriate machine-learning systems for categorizing temporal events collected from web data. Using a case study involving web articles related to an incident of domestic terrorism, the performance of Naive Bayes, SVM, and neural-network methods at applying temporal group labels across a range of feature set sizes is demonstrated. The results show that while all three systems performed in a satisfactory manner, SVM and Naive Bayes increased in accuracy as the number of features increased, while the neural network peaked at 70 features.

[Nizamani et al., 2013] evaluates a number of machine-learning methods (the ID3 decision-tree algorithm, logistic regression, Naive Bayes and SVM) for the purpose of detecting suspicious emails. As well as developing a terrorism-related email dataset for the purposes of this comparison (including real messages gathered from newsgroups), they develop a feature-selection system that provides consistent improvement on the results of all of the tested classifiers. They report that, for their application, logistic regression and ID3 outperformed the Naive Bayes and SVM classifiers.

Detect terrorist activities

After the 9/11 terrorist attacks, more artificial intelligence research and development projects have focused on facilitating knowledge acquisition, assisting in the formation of terrorism-related knowledge bases, and supporting the processes of analysis and decision making in counterterrorism [Markman et al., 2003]. This subsection presents an overview of these research works.

The University of Arizona's AI Lab developed web-based counterterrorism knowledge portals [Reid et al., 2004]. The main focus of this project is to provide advanced methodologies for analyzing terrorism research, terrorists, and the terrorized groups (victims). This project uses pattern matching algorithms which takes in input a query in natural language from the user and matches the input to one of the questions in their question/answer script, then picks out the appropriate response.

[Popp et al., 2005] proposes a collaborative analysis environment, termed NEMESIS, that utilizes various information technologies to collaborate, evaluate, share, and act on the information faster to detect and prevent terrorist attacks. The two analysis tools integrated within the NEMESIS environment are the ASAM system and the ORA tool. The ASAM system combines the HMM and BN methods to detect terrorist activities and generate global threats. The ORA tool, based on social network analysis, models the information flow within terrorist networks and the evolution of the terrorist networks over time.

In [Vu et al., 2007] the authors studied the Timely Energy-efficient k-Watching Event Detection problem (TEKWED) for composite event detection and alarming in Wireless Sensor Networks (WSNs). In order to solve this problem, they proposed a novel scheme that is able to detect events and deliver timely warnings in WSNs. Based on this scheme, an algorithm that considers topology of the network and routing capabilities is proposed. This algorithm builds a set of detection strategies that have several advantages, including the short notification time, energy conservation, and tunable quality of surveillance requirements in event alarming applications. Another statistical event detection approach based on statistical signal processing techniques is proposed in [Gupchup et al., 2009]. In this work, the authors used Principal Component Analysis (PCA) technique to build a compact model of the observed phenomena that detects various events from the collected measurements in environmental monitoring, such as seasonal trends or rain events. The authors use the divergence between actual collected measurements and model predictions to detect the existence of discrete events within the collected data streams.

Five U.K. universities launched DScent, a joint project that "combines research theories in the disciplines of computational inference, forensic psychology and expert decision-making in the area of counterterrorism" and it includes the use of neural networks to identify deceptive behavior of terrorists with an average of 60% success rate [Dixon et al., 2011a].

In [Curtis et al., 2016] the authors are concerned with alleviating the burden of an operator that constantly monitors several video feeds to detect suspicious activities around a secured critical infrastructure. The automated solution proposed in this wok extracts the objects of interest (i.e., car, person, bird, ship) from the image using an iteratively updated background subtraction method, then the object is classified by an artificial neural network (ANN) coupled to a temporal Bayesian filter. The next step is determining the behavior of the object, e.g., entering a restricted zone or stopping and dropping an object. Relevant alerts are issued to the operator should a suspicious event be identified. The authors tried their approach in the automated monitoring of a dumpster, a doorway and a port.

To counter piracy attempts, maritime operators need to quickly and effectively allocate some mobile resources (defender units) to assist a target given the available information about the attackers. In [De Simio et al., 2016] the authors introduce a Decision Support System (DSS) to that end. The DSS has been designed using Game Theory in order to handle the attractiveness of targets and model strategies for attackers and defenders. Game Theory has proved to be a robust tool to identify the best strategy for the defenders given the information and capabilities of opponents. In the proposed framework, the optimal strategy is modeled as the equilibrium of a time-varying Bayesian-Stackelberg game.

[Rao, 2016] elaborates on an architectural approach for designing composable, multiservice and joint wargames that can meet the requirements of several military establishments. This architecture is realized by the design and development of common components that are reused across applications and variable components that are customizable to different training establishments' training simulators. Some of the important Computational Intelligence (CI) techniques (such as fuzzy cognitive maps, game trees, case-based reasoning, genetic algorithms and fuzzy rule-based systems) that are used to design these wargame components are explained with suitable examples, followed by their applications to two specific cases of Joint Warfare Simulation System and an Integrated Air Defence Simulation System for air-land battles.

Text mining techniques are important for security and defense applications since they allow detecting possible threats to security and public safety (such as mentions of terrorist activities or extremist/radical texts). [Inkpen, 2016] discusses information extraction techniques from social media texts (Twitter in particular) and showcases two applications that make use of these techniques: (1) extracting the locations mentioned in tweets and (2) inferring the users' location based on all the tweets generated by each user. The former task is accomplished via a sequence-based classifier followed by disambiguation rules whereas the latter is tackled through deep neural networks.

3.2. Investigated approaches

In this subsection a literature overview of the specific AI approaches adopting in this work thesis are presented. In particular we put the focus on:

- The HHMs approach (adopted in section 5 for the development of STR component)
- The clustering approach (adopted in section 6 for the development of LTR component)
- The CEP approach (adopted in section 7 for the development of MTR component).

3.2.1. **HMM**

The term asymmetric threat refers to tactics employed by, e.g., terrorist groups to carry out attacks on a superior opponent, while trying to avoid direct confrontation. Terrorist groups are elusive, secretive, amorphously structured decentralized entities that often appear unconnected. Analysis of prior terrorist attacks suggests that a high magnitude terrorist attack requires certain enabling events to take place.

In previous works HMMs have been shown to provide powerful statistical techniques, and they have been applied to various problems such as speech recognition, DNA sequence analysis, robot control, fault diagnosis, and signal detection, to name a few. Excellent tutorials about HMMs can be found in [Norris, 1998], [Dymarski, 2011] while a compressive mathematical definition can be gathered in section 5.3.

The basic motivation for modeling terrorist activities via HMMs is twofold. Firstly, carrying out a terrorist activity requires planning and preparations, following steps that form a pattern. This pattern of actions can be modeled using a Markov chain. Secondly, the terrorists leave detectable clues about these enabling events in the observation space. The clues are not direct observations of the planning and preparations, but are rather related to them, meaning that the states in the Markov model are hidden. For example, an observation of a purchase of chemicals could be indicative of intentions to produce a chemical weapon. However, a purchase of chemicals could very well be a benign event, which motivates inclusion of a model of observations that are unrelated to the HMM. The applicability of HMMs for terrorist activity modeling and other national security problem situations has been illustrated in previous work.

[Schrodt, 2000] uses "hidden Markov models" to measure similarities among international crises. The models are first estimated using the Behavioral Correlates of War data set of historical crises, then applied to an event data set covering political behavior in the contemporary Middle East for the period April 1979 through February 1997. A split-sample test of the hidden Markov models perfectly differentiates crises involving war from those not involving war in the cases used to estimate the models. The models also provide a high level of discrimination in a set of test cases not used in the estimated, and most of the erroneously-classified cases have plausible distinguishing features. The difference between the “war” and “nonwar” models also correlates significantly with a scaled measure of conflict in the contemporary Middle East.

[Singh et al., 2004] develops a tool to detect and track terrorist activity. Authors follow two probabilistic approaches: HMMs and Bayesian networks (BNs). Authors assert that HMMs, which are used for modeling partially observed stochastic processes, are an ideal way to make inferences about the evolution of terrorist networks. The HMMs detect the monitored terrorist activity and measure threat levels, whereas BNs combine the likelihoods from many different HMMs to evaluate the cumulative probability of terrorist activity. In other words, BNs represent the overarching terrorist plot and the HMMs, which are related to each BN node, represent detailed terrorist subplots. A case study for the 2004 Olympics is presented in this paper as an example.

The authors of [Coffman and Marcus, 2004] present the methodology and results of a study that applies HMMs to time-varying social network analysis metric values, in order to classify the evolution of simulated social networks. They motivate and present results from a case study using a simulation of suspicious groups communicating in a normal background population. They achieved 96% classification accuracy on novel synthetic data using two 35-state univariate HMMs trained to model normal and suspicious evolutions of the characteristic path length metric.

In this work [Weinstein et al., 2009], the authors describe an approach and some initial results on modeling, detection, and tracking of terrorist groups and their intents based on multimedia data. In particular it describes the development and application of a new Terror Attack Description Language (TADL), which is used as a basis for modeling and simulation of terrorist attacks. Examples are shown which illustrate the use of TADL and a companion simulator based on a HMM structure to generate transactions for attack scenarios drawn from real events. They also describe the techniques for generating realistic background clutter traffic to enable experiments to estimate performance in the presence of mix of data.

In [Singh et al., 2009], the authors introduced feature-aided tracking combined with HMMs for analyzing asymmetric threats. HMMs can detect, track, and predict the potential threat activities in the presence of partial and imperfect sequential data. The proposed approach can also serve as a what-if analysis tool by allowing users to modify models (i.e., states in the HMMs) and/or transaction sequences. The authors utilized a transaction-based probabilistic method to detect and track a pattern consistent with the Development of a Nuclear Weapons Program (DNWP). The simulation results demonstrate that the developed Hidden Markov Models and Feature-Aided tracking method (HMMFA) is an effective method to track asymmetric threats with high accuracy. Performance analysis shows that the detection of HMMs improve with increase in the number of states in an HMM. They have also provided a detailed performance comparison between the HMMFA method and the ML-based data mining method for all the HMMs in the DNWP model.

The authors of [Andersson and Johansson, 2010] propose a two-stage method based on multiple distributed sensors for detection of piracy operations at sea. The proposed method is based on fusion of evidence from radar and optical sensors as well as Automated Identification System (AIS) signals. In the first stage, the sensors perform detection, tracking and classification locally. The outputs act as input to the second stage which performs high-level fusion and disambiguation of the first-stage information. The high-level fusion is performed by HMMs. The reported results show that this approach is able to detect piracy operation at an early stage, i.e. close to the time, or possibly before, the attack has occurred.

[Raghavan et al., 2013] develops a HMM framework to model the activity profile of terrorist groups. Key to this development is the hypothesis that the current activity of the group can be captured completely by certain states/attributes of the group, instead of the entire past history of the group. In the simplest example of the proposed framework, the group's activity is captured by a 2 state HMM with the states reflecting a low state of activity (Inactive) and a high state of activity (Active), respectively. In either state, the days of activity are modeled as a discrete-time Poisson point process with a hurdle-based geometric model being a good fit for the number of attacks per day. While more general models can be considered, even the simplest framework is sufficient for detecting spurts and downfalls in the activity profile of many groups of interest. Their results show that the HMM approach provides a competent alternate modeling framework to the Threshold Auto-Regressive (TAR) model [Enders and Sandler, 2002] and Self-Exciting Hurdle Model (SEHM) approaches [Porter and White, 2012], both in terms of explanatory and predictive powers.

HMMs used in [Granstrom et al., 2015] for modeling asymmetric threats. The observations generated by such HMMs are generally cluttered with observations that are not related to the HMM. In this works the authors proposed a Bernoulli filter which processes cluttered observations and is capable of detecting if there is an HMM present, and if so, estimate the state of the HMM. The presented results show that the proposed filter is capable of detecting and estimating an HMM except in circumstances where the probability of observing the HMM is lower than the probability of receiving a clutter observation.

[Shahir et al., 2015] is focused on the monitoring activity of critical infrastructures like sea lanes, ports, offshore structures (like oil and gas rigs) in order to prevent illegal activities (i.e. smuggling of drugs and weapons, human trafficking), piracy and terrorist attacks. The authors propose a novel situation analysis approach to analyze marine traffic data and differentiate various scenarios of vessel engagement for the purpose of detecting anomalies of interest for marine vessels that operate over some period of time in relative proximity to each other. They consider such scenarios as probabilistic processes and analyze complex vessel trajectories using machine learning to model common patterns. Specifically, in this work the common patterns are represented through HMMs and such patterns are classified using Support Vector Machines. To differentiate suspicious activities from unobjectionable behavior, the authors explore fusion of data and information, including kinematic features, geospatial features, contextual information and maritime domain knowledge. The reported experimental evaluation shows the effectiveness of the proposed approach using comprehensive real-world vessel tracking data from coastal waters of North America.

The HMMs approach used in this thesis (see section 5.2) differs from those proposed in the above ones at a knowledge representation level. The proposed one is richer and more realistic, in fact it takes into account the context information and classifies event based on their relevance and domain information.

3.2.2. **Clustering**

During the last decade different computational approaches has been proposed both for improving both data collection (structured or unstructured data analysis) and for the analysis of that data through the construction of different machine learning models with the aim to forecast long term activities of some terrorist groups or to suggest strategies against terrorists analyzing their past behaviors. In these way all these approaches have the same objectives of those of the LTR prediction activity. Following the literature analysis we could classify all these computational approaches in three principal macro categories mainly based on the approach used and on the objective of the analysis. Starting from the "statistical modelling" related approaches, with particular focus on those having the aim of estimating the risk of particular events, the attention is focalized on "algorithmic approaches" (both supervised and unsupervised) and their link with network based approaches.

The first category, the statistical modelling approaches, includes conventional time series approaches [Pevehouse and Brozek, 2010] and, more recently, a substantial amount of work using vector auto-regression [Brand and Williams, 2007] for conflict early warning. A different approach is that proposed in [Hosmer et al., 2008] that, rather than trying to predict the value of a behavior at a specific time, tries to predict probabilities of events, and in particular analyses the change in probability over time of the next event occurring. This kind of approaches suffers of different data quality related problems and in particular of the so called zero (non-event) observations in terrorism-event data sets. Some other approaches based on "rare events analysis" has been investigated in [McMorrow, 2009] and [King and Zeng, 2001]. Particular attention must be given to the approach proposed in [Clauset and Woodard, 2013] which goal is to estimate the probability of observing at least one "catastrophic" event of size x (where x is the rarest event in the dataset) in an empirical sample. The proposed method provides an objective estimate of the historical or future probability of a rare event, for example, an event that has occurred exactly once.

The second category, the "algorithmic" approaches have proven effective and particularly relevant to forecasting terroristic events. As stated in [Schrodt et al., 2013], these models are less dependent on rigid assumptions about the data generating process and underlying distributions, like the previous ones. In this way algorithmic approaches try to be similar to an human analysts or a decision maker. The vast majority of quantitative studies related to terroristic event employ data that meet the requirements of supervised algorithms, between all we could focalize the attention on linear models, neural networks and Tree-Based Algorithms [Hastie et al., 2009]. Although most works with neural networks seems far removed from terrorism, in [Dixon et al., 2011b] the authors chose the neural network approach in order to distinguish terrorist and non-terrorist behaviors using data generated by a game designed by psychologists and criminologists.

Unlike supervised learning algorithms that train a model based on relationships between a matrix of covariates and a corresponding vector of observed dependent variables for each observation, unsupervised approaches are applied to datasets for which dependent variables are 'latent' and therefore not directly provided. As already underlined, in the domain of terroristic events analysis and forecast one of the important task is to reduce the dimensionality of the space in order to reduce computation and in particular avoid redundant data elements. Analyzing the literature the unsupervised approaches range from factor analysis to reduce the dimensionality of matrix of potential covariates by identifying latent attributes, like Principal Component Analysis (PCA) [Ghahramani, 2004], to clustering approaches (e.g. k-means methods, mixture models, and kernel methods) similar to dimension reduction in that they attempt to identify latent classes amongst a set of observations, but differ in that they identify discrete, rather than continuous solutions like PCA. A benefit of clustering approaches is that features are similar to measured data making the results easier to interpret, however, the binary assignments reduce flexibility. Analyzing unsupervised approaches applied in this domain, k-means approaches are among the most commonly used clustering algorithms. [Malathi et al., 2011] uses k-means clustering with some enhancements to aid in the process of identification of crime patterns. While K-means algorithm is used in particular for the analysis of database in which each feature is described thru numerical attributes, another frequently used approach, it's the CLOPE algorithm for categorical attributes analysis [Yang et al., 2002], [Mahmood and Rohail, 2012].

A major drawback of k -means is that it cannot separate clusters that are non-linearly separable in input space. Two recent approaches have emerged for tackling such a problem. One is kernel k-means, where, before clustering, points are mapped to a higher-dimensional feature space using a nonlinear function, and then kernel k-means partitions the points by linear separators in the new space. The other approach is spectral clustering algorithms, which use the eigenvectors of an affinity matrix to obtain a clustering of the data [Von Luxburg, 2007]. Another kind of approaches are those proposed by [D'Orazio et al., 2011] who suggest an archetype-driven approach to sequence analysis with the explicit goal of predicting political violence that uses sequence analysis to build features on which a statistical or algorithmic model produces out-of-sample forecasts. For this archetype approach be succeed, it requires the existence of a distinct pattern of events (i.e. an archetype) that tends to precede a given outcome-of-interest. If such an archetype exists, then a model may be able to forecast the outcome of interest based on the extent to which a sequence whose outcome is unknown (for example, events being observed in real time) is similar to the archetype sequence. An approach that include both the basis of archetype-driven approaches and unsupervised approaches, is that proposed in [Martinez et al., 2008] and [Xue et al., 2011]. [Martinez et al., 2008] uses the so called CONVEX and SitCAST algorithms to predict future actions of terroristic groups assuming normal behavior. The authors based their study on the database MAROB (see section 2.5.2). More in detail, the algorithm views each instance of as a pair of two vectors: context vector and action vector. The first contains the values of the context variables associated with the terrorist group, while the other one the values of the action variables. In this way the algorithm describes an instance-based learning process (e.g., k-nearest neighbors) used to identify k most similar contexts to the query context based on a historical dataset and predict, through the so called SitCAST algorithm, future behaviors using action features associated with context features.

In the last category of computational approaches, the Network Approaches, we could find works with different goals, like forecast source information in a transnational terroristic network or find hidden links between different organizations. Important work proposed by [Desmarais and Cranmer, 2013] analyzes data by coding nationalities of terrorists and the location of their attack, develops an approach to forecasting the network of transnational terrorist attacks in order to forecast source information. A more complex approach is proposed by [Dawoud et al., 2013], in which is introduced a novel framework which integrates network methods and data mining techniques to model and analyze terrorism networks. First the algorithm builds the terror/criminal network searching the co-occurrences of some keywords which are provided by a domain expert. Then, it partitions it into sub-networks by considering different link types and studies them in order to find hidden links between terroristic groups. The application presented in this kind of approaches are outside the objectives of our LTR prediction task (section 6.2), but are interesting in order to better capture the hidden relations between terroristic events.

3.2.3. C**EP**

As this thesis aims at producing timely proactive responses by relying on reliable forecasts about threat actions, one can see a direct correspondence to event processing systems. The overall field of event processing refers to a broad area with applications in environmental monitoring, monitoring for transportation and logistics, trading for financial markets, security application for intrusion detection, bio-hazard attacks etc. Since the role of Medium Term Reasoning module in the developed framework is to process numerous incoming events types of different nature and update the sensitivity of the STR in near real-time manner, we believe that event processing approaches fits requirements of the MTR module.

In [Etzion and Niblett, 2010] an event is defined as "an occurrence of something that has happened, or is considered as having happened". Events can be primitive (atomic) events (e.g. sensor data, water level, trading ticks, credit card transaction, a mouse click, network signals) and complex (composite) events (e.g. flood, landslide, terrorist attack, plane landing, intrusion detected, credit card fraud). While primitive events don't contain other events, complex events are composed of other primitive and/or complex events.

In an event processing systems continuous event stream are received from different event sources (e.g. sensors, software applications). As data sources (e.g. temperature sensors) have become relatively cheap and are able to produce thousands of measurements every day, the need to continuously process incoming data streams in near real-time has become crucial leading to development and evolution of new tools. In [Cugola and Margara, 2012] the authors propose an abstract framework for event processing (information flow processing) systems that are able to manage multiple data stream sources and derive new information about the data stream through use of a set of processing rules. Two main types of existing information flow processing systems are defined: Data Stream Management Systems (DSMS) and Complex Event Management Systems (CEP). On one hand DSMS are rooted in classical data base management systems (DBMS) they deal with constantly updating data-streams and continuously execute queries as new data arrives. Similarly to DBMSs, they process incoming data through a sequence of transformations based on common SQL-like operators and continuously update the results. On the other hand CEP systems filter and combine incoming events of particular patterns from the external world to understand what high-level complex events have occurred and notify relevant actors or be reused as an input in the CEP solution.

As event patterns must specify complex relationships among input events entering the system they can rely on two types of languages for this [Etzion and Niblett, 2010]: stream-oriented style and rule-oriented style language. The stream-oriented (transforming) style is inspired by SQL and relational algebra (e.g. CQL [Arasu et al., 2006] from the STREAM project) and is used in DSMSs. The rule-oriented (detecting) style, commonly used in CEP systems, defines detecting rules by separately specifying the firing conditions (event patterns) and the actions to be taken when such conditions are satisfied (e.g. event-condition-action (ECA) rules [McCarthy and Dayal, 1989], Snoop rules [Chakravarthy and Mishra, 1994]).

Neither of the above mentioned language solutions can satisfy both the expressivity and effectiveness needs of CEP applications on their own. Hence, alternative CEP languages have been proposed that combine and extend operators from both language styles as in [Wu et al., 2006] and [Wang et al., 2009]. Promising research directions consider using both background knowledge to reason about the events (e.g. ETALIS [Anicic et al., 2011] a logic-rules based CEP that uses contextual knowledge and defines semantic relations between events) and statistical knowledge to detect patterns of interest in event streams.

Even though a crucial feature in CEP systems is real-time processing in order to assure timely reaction, in a certain number of applications (e.g. terrorist threat, credit card fraud) the importance is on proactively preventing events before they occur and not only reacting after they happen. The value of the detected/predicted complex events decreases with time (terrorist threat notification in near real-time as opposed to a day after) as described in [Fülöp et al., 2012] and shown in Figure 8. The described setting satisfied the requirements of the thesis domain and its need to proactively detect and react to terrorist threats. Hence to address this issue we jointly consider both CEP systems and predictive analytics approaches, in this way enabling processing online streams of events while inferring decisions based on past and current data concerning prediction of future events of interest. We accomplish this by learning predictions from both long-term and short-term historical data and integrating the mentioned predictive analytics approaches with real–time complex event processing, the mentioned combination is particularly useful in application where a certain level of uncertainty regarding complex events is allowed.

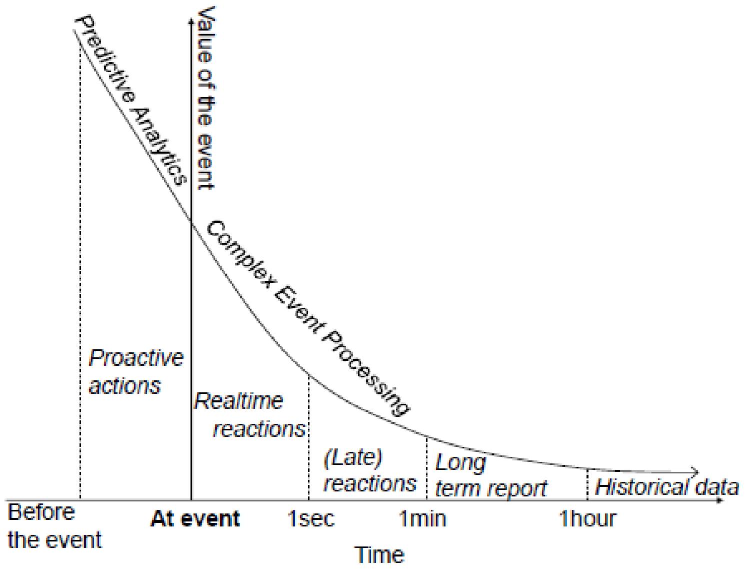

Figure 8 Knowledge value of complex events for an online event stream processing system [Fülöp et al., 2012].

3.3. Software tools and technologies

This section is aimed at presenting the current state of the art about technologies and tools for the analysis of big and stream data. Some of the following tools and technologies were developed concurrently with this thesis, so they have not been used in the development of TENSOR framework. However This overview is particularly relevant in order to provide a more precise positioning of the TENSOR proposed approach and to envisage which are the most suitable solutions for its future extension and development, in particular with respect to the overall architecture.

3.3.1. **Big Data Mining**

In 1997, Cox and Ellsworth [Landset et al., 2015] were among the first authors in scientific literature to discuss big data in the context of modern computing. Their work focused on data visualization, but their observations about the big data problem can easily be extrapolated to general data analytics and machine learning. The big data problem, according to them, consists of two distinct issues:

- **Big data collections** are aggregates of multiple datasets that are individually manageable, but as a group are too large to fit on disk. The datasets in these collections typically come from different sources, are in disparate formats, and are stored in separate physical sites and in different types of repositories.
- **Big data objects** are individual datasets that by themselves are too large to be processed by standard algorithms on available hardware. Unlike collections, they typically come from a single source.

Today, the problem of big data collections is often solved through distributed storage systems, which are designed to carefully control access and manage in a fault-tolerant manner.

The Big Data phenomenon is intrinsically related to the open source software revolution. Large companies such as Facebook, Yahoo!, Twitter, LinkedIn benefit and contribute to open source projects. Big Data infrastructure deals with Hadoop, and other related software as [Fan et al., 2013]:

- **Apache Hadoop**: software for data-intensive distributed applications, is based on MapReduce programming model and a distributed file system called Hadoop Distributed Filesystem (HDFS). Hadoop allows writing applications that rapidly process large amounts of data in parallel on large clusters of compute nodes. A MapReduce job divides the input dataset into independent subsets that are processed by map tasks in parallel. This step of mapping is then followed by a step of reducing tasks. These reduce tasks use the output of the maps to obtain the final result of the job.
- Apache Hadoop related projects: Apache Pig, Apache Hive, Apache HBase, Apache ZooKeeper, Apache Cassandra, Cascading, Scribe and many others.
- **Apache S4**: platform for processing continuous data streams. S4 is designed specifically for managing data streams. S4 apps are designed combining streams and processing elements in real time.
- **Storm**: software for streaming data-intensive distributed applications, similar to S4, and developed at Twitter.

In Big Data Mining, there are many open source initiatives. The most popular are the following:

- **Apache Mahout**: Scalable machine learning and data mining open source software based mainly in Hadoop. It has implementations of a wide range of machine learning and data mining algorithms: clustering, classification, collaborative filtering and frequent pattern mining.
- **R**: open source programming language and software environment designed for statistical computing and visualization. R was designed by Ross Ihaka and Robert Gentleman at the University of Auckland, New Zealand, beginning in 1993 and is used for statistical analysis of very large data sets.
- **MOA (Massive Online Analysis)**: Stream data mining open source software to perform data mining in real time. It has implementations of classification, regression, clustering and frequent item set mining and frequent graph mining. It started as a project of the Machine Learning group of University of Waikato, New Zealand, famous for the **WEKA (Waikato Environment for Knowledge Analysis)** software. The streams framework provides an environment for defining and running stream processes using simple XML based definitions and is able to use MOA, Android and Storm. **SAMOA (Scalable Advanced Massive Online Analysis)** is a new upcoming software project for distributed stream mining that will combine S4 and Storm with MOA.
- **Vowpal Wabbit**: open source project started at Yahoo! Research and continuing at Microsoft Research to design a fast, scalable, useful learning algorithm. VW is able to learn from terafeature datasets. It can exceed the throughput of any single machine network interface when doing linear learning, via parallel learning.

There are many future important challenges in Big Data management and analytics that arise from the nature of data: large, diverse, and evolving [Fan et al., 2013]. These are some of the challenges that researchers and practitioners will have to deal with in the years to come:

- **Analytics Architecture**. It is not clear yet how an optimal architecture of an analytics systems should be constructed to deal with historic data and with real-time data at the same time. An interesting proposal is the Lambda architecture of Nathan Marz. The Lambda Architecture solves the problem of computing arbitrary functions on arbitrary data in real-time by decomposing the problem into three layers: the batch layer, the serving layer, and the speed layer. It combines in the same system as Hadoop for the batch layer, and Storm for the speed layer. The properties of the system are: robust and fault tolerant, scalable, general, extensible, allows ad hoc queries, minimal maintenance, and debuggable.
- **Evaluation**. It is important to achieve significant statistical results, and not be fooled by randomness. As Efron explains in his book about Large Scale Inference, it is easy to go wrong with huge data sets and thousands of questions to answer at once. Also, it will be important to avoid the trap of a focus on error or speed as Kiri Wagstaff discusses in her paper “Machine Learning that Matters”.
- **Distributed mining**. Many data mining techniques are not trivial to paralyze. To have distributed versions of some methods, a lot of research is needed with practical and theoretical analysis to provide new methods.

- **Time evolving data**. Data may be evolving over time, so it is important that the Big Data mining techniques should be able to adapt and in some cases to detect change first. For example, the data stream mining field has very powerful techniques for this task.
- **Compression**. Dealing with Big Data, the quantity of space needed to store it is very relevant. There are two main approaches: compression where we don't lose anything, or sampling where we choose data that is more representative. Using compression, we may take more time and less space, so we can consider it as a transformation from time to space. Using sampling, we are losing information, but the gains in space may be in orders of magnitude.
- **Visualization**. A main task of Big Data analysis is how to visualize the results. As the data is so big, it is very difficult to find user-friendly visualizations. New techniques, and frameworks to tell and show stories will be needed, as for example the photographs, infographics and essays in the beautiful book "The Human Face of Big Data".
- **Hidden Big Data**. Large quantities of useful data are getting lost since new data is largely untagged file based and unstructured data. The 2012 IDC study on Big Data explains that in 2012, 23% (643 Exabytes) of the digital universe would be useful for Big Data if tagged and analyzed. However, currently only 3% of the potentially useful data is tagged, and even less is analyzed.

3.3.2. Machine Learning for Big and Streaming Data

One solution for the problem of big data objects in machine learning is through parallelization of algorithms. This is typically accomplished in one of two ways [Bekkerman et al., 2011]:

- **Data parallelism**, in which the data is divided into more manageable pieces and each subset is computed simultaneously,
- **Task parallelism**, in which the algorithm is divided into steps that can be performed concurrently.

While Hadoop is ubiquitous as a big data framework, there are a number of other open source options for machine learning that do not use it at all. **MOA** is a WEKA-related project offering online stream analysis on a number of (WEKA) algorithms and with the same user interface. **MADlib** is a collection of SQL-based algorithms designed to run at scale within the database rather than porting data between multiple runtime environments. It includes clustering, classification, regression, and topic models as well as tools for validation. **Dato**, formerly **GraphLab**, is a standalone product that can be connected with Hadoop for graph analysis and Machine Learning tasks. It was fully open source, but in late 2014, they transitioned into a commercial product. Their C++ processing engine **Dato Core** has been released to the community on Github along with their interprocess communication library (for translating between C++ and Python) and graph analytics implementations. Their machine learning libraries are unavailable outside of their enterprise packages. Distributed processing on Hadoop enables large-scale learning, while non-distributed tools for machine learning are widely available, and are thus more mature for use in projects that do not handle Big Data.

Table 3 Data processing engines for Hadoop [Landset et al., 2015].

	Current stable release (as of June 1, 2015)	Execution model	Supported languages	Associated ML tools	In-memory processing	Low latency	Fault tolerance	Enterprise support
MapReduce	2.7.0	Batch	Java	Mahout	X	X	✓	X
Spark	1.3.1	Batch, streaming	Java, Python, R, Scala	MLib, Mahout, H_2O	✓	✓	✓	✓
Flink	0.8.1	Batch, streaming	Java, Scala	Flink-ML, SAMOA	✓	✓	✓	X
Storm	0.9.4	Streaming	Any	SAMOA	✓	✓	✓	X
H_2O	3.0.0.12	Batch	Java, Python, R, Scala	H_2O, Mahout, MLib	✓	✓	✓	✓

The MapReduce approach to Machine Learning performs batch learning, in which the training data set is read in its entirety to build a learning model. The biggest drawback to this batch model is a lack of efficiency in terms of speed and computational resources. In a typical batch-oriented workflow, the set of training data is read from the HDFS to the mapper as a set of key-value pairs. The output, a list of keys and their associated values, is written to disk. In a classification task, for example, the initial key-value pair might be a filename and a list of instances, and the intermediate output from the mapper would be a list of each instance with its associated class. This intermediate data is then read into one or more reducers to train a model based on this list.

These frequent I/O operations can become very expensive in terms of time, computational resources, and network bandwidth. Any model parameters that need to be tuned after the initial evaluation stage further add to the costs. These issues become more apparent in cases where it is necessary to update models with changing data, which is often the case in real-world Machine Learning production environments. While this approach may be suitable for certain projects such as analyzing past events, it becomes problematic when data evolves, as the full process must be repeated each time a model requires updating.

Spark is an Apache top-level project based on MapReduce but addresses a number of the deficiencies described above. It supports iterative computation and it improves on speed and resource issues by utilizing in-memory computation. Spark’s approach to processing has seen widespread adoption in both research and industry. The main abstractions used in this project are called Resilient Distributed Datasets (RDD), which store data in-memory and provide fault tolerance without replication. RDDs can be understood as read-only distributed shared memory. This model streamlines the learning process through in-memory caching of intermediate results, significantly cutting down on the number of read and write operations necessary. The RDD API was extended in 2015 to include DataFrames, which allow users to group a distributed collection of data by column, similar to a table in a relational database. Spark is easy to program and part of that reason is due to the fact that it can be coded in Java, R, Python, or Scala. For machine learning tasks, Spark ships with the **MLlib** and **GraphX** libraries and the latest version of the **Mahout** library offers a number of Spark implementations as well.

Storm is used for processing data in real-time and was initially conceived to overcome deficiencies of other processors in collecting and analyzing social media streams. Development on Storm began at BackType, a social media analytics company and continued at Twitter after a 2011 acquisition. The project was open sourced and became an Apache top-level project in September 2014. The machine learning community has been placing growing importance on real-time processing, and as a result, Storm is seeing increased adoption both in production and in research environments. The Storm architecture consists of spouts and bolts. A spout is the input stream (e.g. Twitter streaming API), while bolts contain most of the computation logic, processing data in the form of tuples from either the spout or other bolts. Networks of spouts and bolts, which are represented as directed graphs, are known as topologies. Storm was built as a stand-alone system independent from Hadoop, but since Hadoop moved to YARN, work has been done to integrate the two projects. Hortonworks added Storm to their Hadoop distribution beginning in version 2.1 and Yahoo! is working on an integration as well. The principal developer of Storm, Nathan Marz, coined the term **"Lambda Architecture"**, describing a generalized approach to combine multiple paradigms into one system by breaking down processing into three layers: **batch**, **serving**, and **speed**. The **batch layer** stores the master dataset and computes views which are sent to the serving layer for indexing and keeping track of the most current results. The **speed layer** looks at new data only, as it arrives, and makes updates in real-time. New data is sent to both the batch layer and the speed layer for computation and results from each are merged when the system is queried. In terms of the processing engines we have discussed so far, the lambda architecture can be seen as a way to quickly run jobs on MapReduce and Storm simultaneously and combine the results. This unifies the processing of both real-time and historical data. Storm does not ship with a Machine Learning library, but **SAMOA**, a platform for mining big data streams, currently has implementations for classification and clustering algorithms running on Storm. **H2O** has also offered a way to link the two projects.

Flink graduated the Apache incubation stage in January 2015 and is now a top level project. It offers capability for both batch and stream processing, thus allowing for the implementation of a Lambda Architecture as described above. It is a scalable, in-memory option that has APIs for both Java and Scala. It has its own runtime, rather than being built on top of MapReduce. As such, it can be integrated with HDFS and YARN, or run completely independent from the Hadoop ecosystem. Flink's processing model applies transformations to parallel data collections. Such transformations generalize map and reduce functions, as well as functions such as join, group, and iterate. Also included is a cost-based optimizer which automatically selects the best execution strategy for each job. Flink is also fully compatible with MapReduce, meaning it can run legacy code with no modifications. Like Spark, Flink also offers iterative batch as well as streaming options, though their streaming API is based on individual events, rather than the micro-batch approach that Spark uses. This is the same model that Storm uses for true real-time processing. Connectors are offered which allow for processing data streams from **Kafka**, **RabbitMQ** (a platform-independent messaging system), **Flume**, Twitter, and user-defined data sources. The project is still in its infancy but machine learning tools are in development. **Flink-ML**, a machine learning library, was introduced in April 2015. Additionally, an adapter is available for the **SAMOA** library.

H2O is an open source framework that provides a parallel processing engine, analytics, math, and machine learning libraries, along with data preprocessing and evaluation tools. Additionally, it offers a web-based user interface, making learning tasks more accessible to analysts and statisticians who may not have strong programming backgrounds. For those who wish to tweak the implementations, it offers support for Java, R, Python, and Scala. In addition to its native processing engine they have also released a project called **Sparkling Water** which integrates **Spark** and **Spark Streaming** into their platform. This is only supported in version 3.0. Additional efforts have been made towards integration with Storm for real-time streaming. H2O's engine processes data completely in-memory using multiple execution methods, depending on what is best for the algorithm used. The general approach used is Distributed Fork/Join, a divide-and-conquer technique, which is reliable and suitable for massively parallel tasks. This is a method which breaks up a job into smaller jobs which run in parallel, resulting in dynamic fine-grain load balancing for MapReduce jobs as well as graphs and streams. They claim to be the fastest execution engine, but as of the time of this writing, no academic studies have been published which verify or refute these claims and further research is needed in this area.

A variety of machine learning toolkits have been created to facilitate the learning process but many researchers and practitioners reject them for various reasons, most often because they lack needed features or are difficult to integrate into an existing environment. One issue is that machine learning is a broad field of study and many of the available toolkits lack important functionality. Another problem is that without true expertise in the areas of programming and system architecture, many people lack a full understanding of what the various platforms are capable of.

Table 4 Overview of machine learning toolkits [Landset et al., 2015].

	Mahout	**MLlib**	**H2O**	**SAMOA**
Interface language	Java	Java, Python, Scala	Java, Python, R, Scala	Java
Associated platform	MapReduce, spark (H2O and flink in progress)	Spark, H_2O	H_2O, Spark, MapReduce	Storm, S4, Samza
Current version (as of June 1, 2015)	0.10.1	1.3.1	3.0.0.12	0.2.0
Graphical user interface	-	-	✓	-
Classification and regression algorithms				
Decision tree	-	✓	-	✓[a]
Logistic regression	✓[b]	✓[a]	✓	-
Naïve Bayes	✓	✓	✓	-
Support vector machine	-	✓	-	-
Gradient boosted trees	-	✓	✓	-
Random forest	✓	✓	✓	-
Adaptive model rules	-	-	-	✓[a]
Generalized linear model	-	-	✓	-
Linear regression	-	✓[a]	✓	-

Clustering algorithms				
k-Means	✓	✓	✓	-
Fuzzy k-means	✓	-	-	-
Streaming k-means	✓	✓[a]	-	-
Power iteration	-	✓	-	-
Spectral clustering	✓	-	-	-
CluStream	-	-	-	✓[a]
Collaborative filtering (cf) algorithms				
User-based CF	✓	-	-	-
Item-based CF	✓	-	-	-
Alternating least squares	✓	✓	-	-
Dimensionality reduction and feature selection tools				
Principal component analysis	✓	✓	✓	-
QR decomposition	✓	-	-	-
Singular value decomposition	✓	✓	-	-
Chi squared	✓	-	-	-
Additional algorithms				
Association rule learning	✓	✓	-	✓[a]
Deep learning	-	-	✓	-
Topic modeling	✓	✓	✓	-

[a] Real-time streaming implementation.
[b] Single machine, trained using Stochastic gradient descent.

3.3.3. Evaluating Machine Learning for Big and Streaming data

Mahout is one of the more well-known tools for Machine Learning. It is known for having a wide selection of robust algorithms, but with inefficient runtimes due to the slow MapReduce engine. In April 2015, Mahout 0.9 was updated to 0.10.0, marking something of a shift in the project's goals. With this release, the focus is now on a math environment called **Samsara**, which includes linear algebra, statistical operations, and data structures. The goal of the **Mahout-Samsara** project is to help users build their own distributed algorithms, rather than simply a library of already-written implementations. They still offer a comprehensive suite of algorithms for MapReduce and many have been optimized for Spark as well. Integrations with **H2O** and **Flink** are currently in development.

The algorithms included in Mahout focus primarily on classification, clustering and collaborative filtering, and have been shown to scale well as the size of the data increases. Additional tools include topic modeling, dimensionality reduction, text vectorization, similarity measures, a math library, and more. One of Mahout's most commonly cited assets is its extensibility and many have achieved good results by building off baseline algorithms. However, in order to take advantage of this flexibility, strong proficiency in Java programming is required. Committer Ted Dunning noted

"It's not a product. It's not a package. It's not a service. Batteries are not included." Some researchers have cited difficulty with configuration or with integrating it into an existing environment. On the other hand, a number of companies have reported success using Mahout in production. Notable examples include Mendeley, LinkedIn, and Overstock.com, who all use its recommendation tools as part of their big data ecosystems. Overstock even replaced a commercial system with it, saving a significant amount of money in the process.

MLlib covers the same range of learning categories as Mahout, and also adds regression models, which Mahout lacks. They also have algorithms for topic modeling and frequent pattern mining. Additional tools include dimensionality reduction, feature extraction and transformation, optimization, and basic statistics. In general, MLlib's reliance on Spark's iterative batch and streaming approaches, as well as its use of in-memory computation, enable jobs to run significantly faster than those using Mahout.

However, the fact that it is tied to Spark may present a problem for those who perform machine learning on multiple platforms. MLlib is still relatively young compared to Mahout. As such, there is not currently an abundance of published case studies that have used this library, and there is very little research providing meaningful evaluation. The research that has been published indicates it is considered to be a relatively easy library to set up and run, helped in large part by the fact that it ships as part of its processing engine, thus avoiding some of the configuration issues people have reported with Mahout.

The documentation is thorough, but the user community is not nearly as active as the community developing for it. This issue is expected to improve as more people are migrating from MapReduce to Spark. The large and active group of developers means that many complaints are fixed before they are even published. Notable examples of companies that use MLlib in production are OpenTable and Spotify, both for their recommendation engines.

For classification, MLlib offers Supports Vector Machines, Logistic Regression, Naïve Bayes, Decision Trees, Random Forest, and Gradient-Boosted Trees. Clustering algorithms include k-Means, Gaussian Mixture, and Power Iteration Clustering. They offer implementations for Linear Regression and Isotonic Regression, and one collaborative filtering algorithm using Alternating Least Squares. For online learning, streaming versions of Logistic Regression, Linear Regression, and k-Means Clustering are included. For all other algorithms, models can be learned offline using historic data and applied online to new streaming data. MLlib includes APIs for development in Scala, Java and Python, but not every tool is available in all languages.

Building machine learning pipelines can be a difficult task, particularly when working with a combination of disparate tools. Spark ML, a set of uniform APIs for creation and tuning of pipelines was introduced in version 1.2 to address these issues, making it easier to combine multiple algorithms into one workflow. This package includes tools for dataset transformations and combining algorithms. It works by representing a pipeline as a sequence of dataset transformations.

An easy example of this is to think of a learner which transforms a DataFrame with features into one with predictions. This package is designed to handle all steps of the learning process, starting with importing data from a source, to extracting features, and training and evaluating models.

H2O is the only one that can be considered a product, rather than a project. While they offer an enterprise edition with two tiers of support, nearly all of their offerings are available open source as well and can be used without the purchase of a license. The most notable features of this product are that it provides a graphical user interface (GUI), and numerous tools for deep neural networks.

Deep learning has shown enormous promise for many areas of machine learning, making it an important feature of H2O. There is another company offering open source implementations for deep learning, Deeplearning4j, but it is targeted towards business instead of research, whereas H2O targets both. Additionally, Deeplearning4j's singular focus is on deep learning, so it doesn't offer any of the other types of ML tools that are in H2O's library. There are also other options for tools with a GUI, such as Weka, KNIME, or RapidMiner, but none of them offer a comprehensive open source machine learning toolkit that is suitable for big data.

Programming in H2O is possible with Java, Python, R and Scala. Users without programming expertise can still utilize this tool via the web-based UI. Because H2O comes as a package with many of the configurations already tuned, set up is easy, requiring less of a learning curve than most other free options. While H2O maintains their own processing engine, they also offer integrations that allow use of their models on Spark and Storm.

As of May 2015, the machine learning tools offered cover a range of tasks, including classification, clustering, generalized linear models, statistical analysis, ensembles, optimization tools, data preprocessing options and deep neural networks. On their roadmap for future implementation are additional algorithms and tools from these categories as well as recommendation and time-series. Additionally, they offer seamless integration with R and R Studio, as well as Sparkling Water for integration with Spark and MLlib. An integration with Mahout is currently in the works as well. They offer thorough documentation and their staff is very communicative, quickly answering questions in their user group and around the web.

SAMOA, a platform for machine learning from streaming data, was originally developed at Yahoo! Labs in Barcelona in 2013 and has been part of the Apache incubator since late 2014. Its name stands for Scalable Advanced Massive Online Analysis. It is a flexible framework that can be run locally or on one of a few stream processing engines, including Storm, S4, and Samza. This is done through a minimal API designed for a general distributed stream processing engine which allows users to easily write bindings to port SAMOA to new stream processors.

Though they currently offer far fewer algorithms, they like to call themselves "Mahout for streaming." SAMOA's algorithms are represented as directed graphs, referred to as topologies (borrowing terminology from Storm). The algorithms implemented so far can be used for classification, clustering, regression, and frequent pattern mining, along with boosting, and bagging for ensemble creation. Additionally, there is a common platform provided for their implementations, as well as a framework for the user to write their own distributed streaming algorithms. It does not yet have an active community, but it offers thorough documentation. This platform is meant for users with very big data that is constantly being updated.

Streaming models are for projects aimed at finding out what is happening right now, and feedback occurs in real-time. SAMOA was designed with the goals of flexibility in updating the library (developing new implementations as well as reusing existing ones from other frameworks), scalability in its handling of an increasing amount of data, and extensibility in terms of the APIs described above. Internal tests have resulted in high speed and accuracy.

So far, there are only a few learning tools implemented in SAMOA, but they cover the many of the common ML tasks. For clustering, they now offer CluStream, and for classification there is the Vertical Hoeffding Tree, which utilizes vertical parallelism on top of the Very Fast Decision Tree, or Hoeffding Tree. This is the standard decision tree algorithm for streaming classification tasks. Regression can be accomplished through the Adaptive Model Rules Regressor, which includes implementations for both vertical and horizontal parallelism. The library also includes Distributed Stream Frequent Itemset Mining. Prequential Evaluation is available as well, which enables measurement of model accuracy, either from the start or based on a sliding window of recent instances. Bagging, Adaptive Bagging, and Boosting can be used to create ensembles of classifiers. For additional learning algorithms, there is a plugin available called SAMOA-MOA which allows the use of MOA classifiers and clustering algorithms inside the SAMOA platform. However, it is important to keep in mind that this does not change the underlying implementations of MOA's algorithms, which are not distributed. SAMOA is a very young project and new tools are continually being developed to expand the library. There is not a great deal of independent research on this platform, though that is likely to change as SAMOA becomes more well-known and online learning becomes more widely used.

4. TENSOR: the proposed architecture

This work thesis illustrates a general analytics layered framework that supply help at different users that work in counter-terrorism domain. A brief remark about different users types and their roles in the system (section 2.1) is outlined before describing the overall framework. We evaluated three different types of users:

- Operational users have the aim to identify and notify tactical users about suspicious situations in a physical environment of limited size and complexity (i.e., a city zone) taking into account short histories of events provided by sensors (human or device).
- Tactical users need to analyze a medium term history (sensor data and threats notification) coming from different zones in order to infer the sensitivity/criticality of the current situation in each monitored zone.
- Strategic users work in order to predict the criticality situation of each monitored zone tacking into account historical data and external data sources.

4.1. Framework overview

The overall aim of the Analytics Framework (AF) is to detect and predict the activities of terrorist groups, analyzing terrorist group previous behaviors and real time information provided by virtual sensors. In this work, the term “virtual sensors” refers to a heterogeneous set of sensors composed by devices (e.g. cameras and their image analysis algorithms correlated) and people (that send information into the system through specific applications/devices) while the term “event” refers to the notification of an abnormal situations provided by virtual sensors.

We approach the design of the framework by taking into account both the abstraction levels of the potential information sources (sensor information, police patrol inputs, news events, external data sources) and the expert user roles that are currently defined as crucial in the intelligence analyst flow for analyzing/detecting potential terrorist threats (section 2.1).

In particular, this framework is designed to provide:

- Easy integration of multiple sensors and fusion algorithms at multiple levels (including JDL levels);
- Support for both low-level rule-based and high-level event-based fusion capabilities;
- A set of general-purpose modules that can be implemented in various ways, including different algorithms and designs;
- Transparent interfaces for processing modules and data sensors for quick integration of third party components;
- Multiple data stream sources management and new information about the data stream extraction;
- Real-time delivery of relevant information in a functioning format;
- Prediction of events before they occur and not only reacting after they happen.

4.1.1. **Sensor data collection**

AF needs to enable processing, combination and fusion of multiple data feed, stemming from heterogeneous sources. Our implementation of the framework architecture is based on open source GSN (Global Sensor Networks) middleware [Ha et al., 2015]. GSN provides a flexible middleware for the gathering and processing of data streams generated from different sensors, based on its virtual sensor concept. According to the description of virtual sensors and their deployment in the GSN middleware, the GSN platform supports:

- Data acquisition from various sensors
- Filtering of data based on SQL syntax
- Execution of customizable algorithms on the query results
- Communication of the generated data.

Moreover, the GSN sensor middleware ensures:

- Support for virtually any type of sensor and data stream
- Flexibly addition of new types of sensor networks without interruption of on-going system operation.
- Support for very large numbers of data producers and consumers with a variety of application requirements.

Finally, the last goal of the data processing sensor middleware is to translate, in a common Event Detection Dictionary (EDD), the different sensors output formats. Since the AF has to be generic and usable for detecting terroristic attacks across multiple scenarios, the EDD takes into account the most common known activities that when are observed, are able to give subjections on terroristic attacks. According to [Bennet, 2007], there are eight families of indicators of future terroristic attacks and relevant security incidents. These families indicate a potential taxonomy of terroristic events, which are used to classify the various events provided by virtual sensors. Table 5 illustrates the event detection dictionary with the eight different categories of terrorist indicators, along with some sample set of indicators for each category.

Table 5 Classification of Terrorist Indicators.

Terrorist Indicator Category	Sample Indicators (examples)
Preoperational Surveillance	• Foot surveillance involving two or three individuals working together. • Mobile surveillance using bicycles, scooters, motorcycles, sport utility vehicles, cars, trucks, boats or small aircraft.
Seeking and Eliciting Information	• Inquiries about size of security force. • Inquiries concerning access to sensitive areas.
Probing and Testing Security Measures	• Initiation of false alarms (e.g., a bomb threat). • Attempts to penetrate physical security barriers.

Intrusion (against physical security or cybersecurity measures)	• An intruder enters a restricted area with malicious intent, damaging or manipulating some system of the target. • Intrusion into a computer network.
Acquiring Supplies	• Suspicious or improper attempts to acquire official vehicles, uniforms, badges, access cards, or identification for key facilities • Theft of two-way radios or scanners.
Identification of Suspicious People	• Persons or vehicles observed in the same location on multiple occasions and/or those who engage in unusual behavior • Persons observed near a potential target using or carrying video, still camera, or visual enhancement devices (telescopes, binoculars, night vision goggles).
Dry Run or Trial Run of an Attack	• Suspicious persons sitting in a parked car for an extended period of time for no apparent reason. • Persons observed monitoring a police radiofrequency and recording emergency response times
Deploying Assets and Getting in Position	• Loading Weapons and other supplies in vehicles • Suspicious Behaviors

4.1.2. **User requirements**

Essential user requirements were collected by mean of recommendations obtained during domain expert users consultations. Main criteria to perform requirements analysis are in accordance with discriminations adopted by Baxter and colleagues [Baxter et al., 2015]:

- User roles;
- User needs;
- User expectations.

As describe in section 2.1, AF must interact with different users who have different needs according to their different roles. During this analysis, taking in consideration different user roles, we tried to collect the user requirements separately, in order to avoid the possibility that users influence each other [Baxter et al., 2015]. However, there are some general requirements which have been expressed by the majority of users. These uses requirements can be summarized:

- System enabling the detect of terrorist attacks regardless specific scenario. This requirement implies that AF must be able to easily adapt to specific monitored area (e.g. a city composite of different zones like squares, parking, etc.).
- System for assisting intelligence analysts to perform tasks in a more efficient way. Users reported that intelligence analysts receive a large amount of data including text, audio, video, images, etc. The framework should play a leading role in decision makers support releasing intelligence information to recognize potential targets.

- Specific user friendly GUIs. Users express the concept that the information reported in the various GUIs must be consistent with the role played by users. Moreover, these GUIs must summarize in an easy way all the information needed by users to make their activities.

This kinds of functionalities correspond to a sort of a decision support system for end uses. This system will help human decision makers (e.g. intelligence analysts) by efficiently capturing, fusing and processing large amounts of multi-sensor information. However, as expressly requested by users, such a system should only propose a possible choice, allowing end users to make final decision.

Subsequently to general user requirements, it's appropriate to analyzes specific requirements for each users role. We start with the operational user, who have expressed the need to develop a system able to:

- Detect potential threat situations in a specific monitored zone. Receiving as input all detected events (from virtual sensors), AF will generate potential threat notifications that will be notified at operational users, providing zone-specific threat detection strategies. In particular, AF has to provide a reasoning component that reduce the information overflow generated by sensors and notifies only events that can be "labelled" as potential symptoms of threat situations.
- Change threat detection strategy according to the alert level of a zone. According to what express in section 2.2, each monitored zone has an associated alert level that summarize the probability to have a terrorist threat in that zone. By this requirement, operational users suggest us that in order to analyze the event stream provide by virtual sensors, the threat detection strategies implemented in the AF, must take in consideration also the alert level of the zone.
- Store events stream and threat notifications for future evaluations. The possibility to consult what happened in a zone is crucial for operational users, especially for establish if the threat notification provided by AF refers to a real threat situation or not. However, the framework must allow each operational user to consult only information related to the specific zone of competence.
- Provide almost a complete prospective of on-going terrorist action, in order to prepare their operational plans. The system will provide the potentialities to detect and analyze activities indicating potential attacks. It contributes to this requirement, inferring the alert level of a specific monitored zone and analyzing its relationships with the other monitored zones.
- Customize threat detection strategies. During operational users interaction, the system needs to collect feedbacks in order to implement a self-adaptive threat detection strategy customization.

Analyzing requirements collected from operational users, it is deducible that their requirements are consistent with their partial view of monitored area. Conversely, according to the different activities carried out by tactical users, it is appropriate to focus on whole monitored area. These users' requirements are related to the alert level selection activity and can be outlined in:

- Provide alert level estimations of each monitored zone. AF will assign a new alert level for each monitored zone tacking into account:
    - A medium term history of events and threat notifications occurred during a time window in the order of 15/20 minutes;
    - The actual alert levels of each monitored zones;
    - The predicted alert level for each types of monitored zone (provided by AF);
    - Eventual intelligence information provided by users.
- Provide a system in which it is possible to specify some spatial constraints among different monitored zones. During the configuration phase of the monitored area, the framework needs to allow tactical user to set some spatial constraints between different monitored zones. This requirement suggested us to take in consideration even some spatial constraints for alert level selection activity.
- Summarize to tactical users all the information needed for alert level selection activity. To realize this it was provided a GUI containing all the above information considered for the alert level selection.

Analyzing carefully tactical requirements, we can note that tactical users require a system that correspond to general requirement for a DSS functionalities.

Finally, we analyzed strategic user needs, which role is mainly focused on predict the criticality of each monitored zone, taking into account historical data and external data sources too. Strategic users, therefore, require to:

- Provide Long-term predictions of potential target type of terrorist actions. Receiving in input all detected events, alert levels for each monitored zones, in addition to information provided by external data sources, AF will suggest as output alert level prediction to adopt for each type of monitored zone.
- Provide a system in which Long-term prediction can be set/updated by users. Also in this case, strategic users require to change the Long-term prediction provided by AF according to their experience. The Long-term GUI needs to enable this requirement.

4.1.3. TENSOR Framework architecture

The architecture of proposed analytics framework, called TENSOR (clusTEriNg terroriSm actiOn pRediction), is depicted in Figure 9. Note that the “event detection task” (i.e. sensors stream analysis) is not addressed by this work thesis, but it is reported only for clarity. The information provided by virtual sensors are input of TENSOR but are provided by external components of the framework.

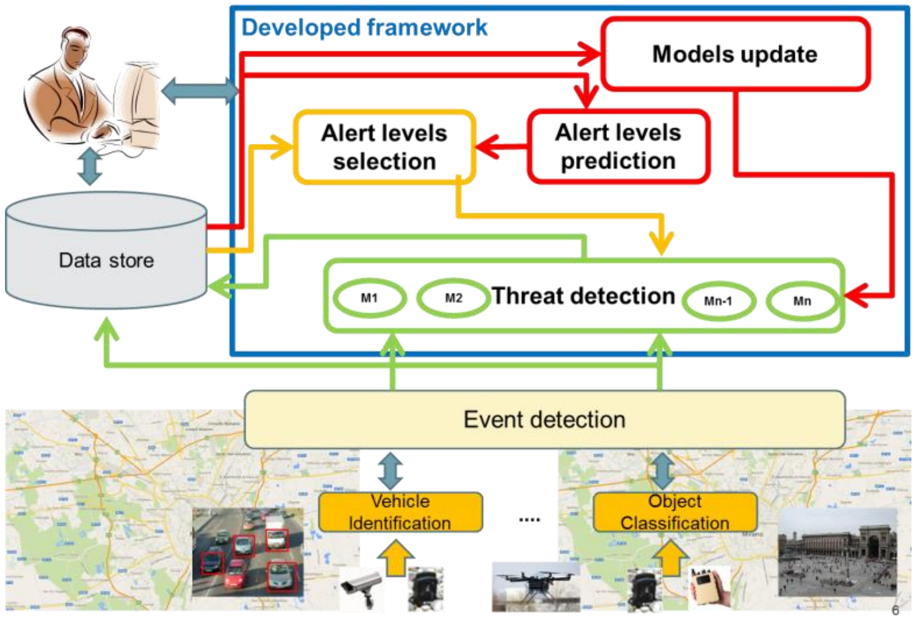

Figure 9 Layered solution approach.

TENSOR is composed of three different modules, that work in different time constrains:

- Short Term Reasoning (STR) (seconds)
- Medium Term Reasoning (MTR) (15/20 minutes)
- Long Term Reasoning (LTR) (few days/ a weak)

The goal of STR is to reduce the information overload of operational users induced by various data sources. The STR implements the capability to aggregate alerts together and to capture patterns that might be related to suspicious situations, therefore reducing the amount and improving the relevance of information that an operational user needs to consider. The STR role is the real time threats detection, based on events occurred in a short time-window (some seconds). All the detected events and all the threat notification are store in a database in order to be used by other modules, MTR and LTR, and users consultations.

MTR assists tactical users on the re-evaluation of the alert level. MTR takes in consideration different information like: (1) the used alert level (from STR), (2) the predicted alert level (from LTR), and (3) the medium-term histories of short-term events and threat notification occurred in a time window of 15/20 minutes.

The LTR analyzes histories of threats and their associated detected events on a long time horizon, in order to help strategic users in the alert levels prediction activity. In particular, LTR provides alert levels predictions for specific types of physical zones (e.g., public buildings, metro stations, and so on). For this activity, LTR module also uses information provided by external sources. Moreover, LTR supplies a “user in the loop” procedure for threat detection modules update.

As shows in Figure 9, during all the TENSOR's activities, there is a direct interaction with the users. However, according to the users requirements, TENSOR proposes numerous choices, leaving the final decision to the users.

4.1.4. **STR - MTR - LTR in the JDL model**

Figure 10 recalls the functionalities of the JDL levels and highlights where each developed module enters the stage. In this figure, it is possible to see that:

- STR is directly involved in the real-time upstream information and processing flow from data sources to end users. However it is located in Level 3, in which the identification and notification of threats falls inside a "man-in-the-loop" scheme, so that the timing constraints are consistent with the decision delays of the users.
- MTR and LTR are in Level 4, and their activities are not bound to strong real-time constraints. Level 4 activities are related to tactical and strategic issues and are executed off-line or in background in respect to the real-time activities of levels 0–3.

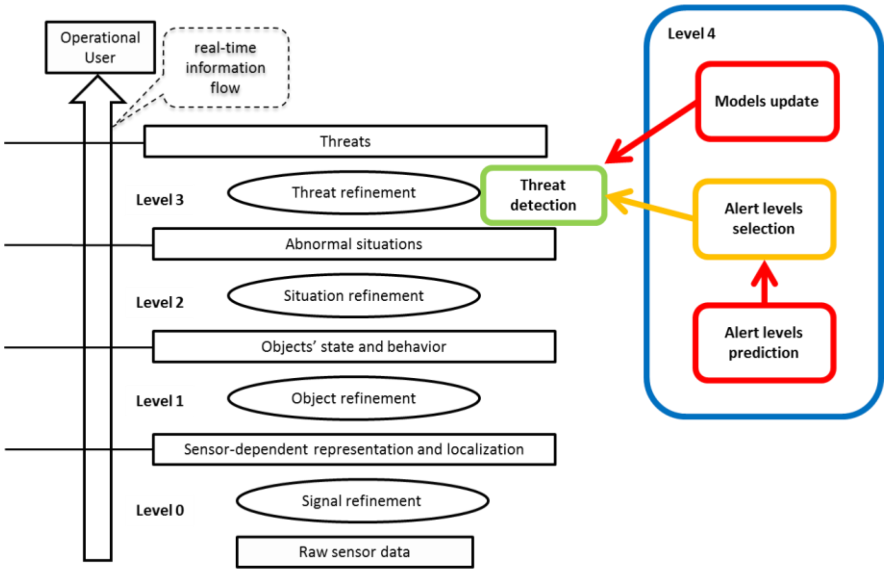

Figure 10 JDL Layers and STR, MTR and LTR .

According to the previous consideration, in Figure 10 it is possible to see that Level 3 is focus on the threat detection activity, while Level 4 is focus on the alert level selection, alert level prediction and modules update activities.

5. Short Term Reasoning layer

The threat detection is a typical short term activity [Petris et al., 2014] aimed at the identification of significant occurrence or pattern that is unusual comparing to the normal patterns of the system. In particular, the TENSOR threat detection component analyzes short term events provided by sensor in order to detect potential threat.

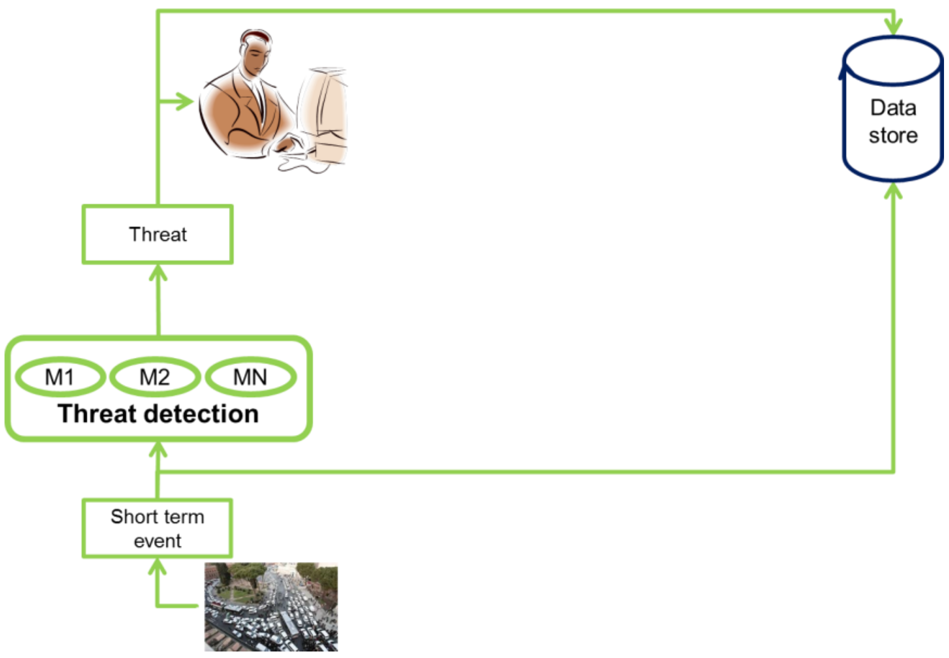

Figure 11 Short Term Reasoning.

As it is possible to see in Figure 11, all the detected events and the threat notifications are stored in order to be reused in the next activities. Note that a threat notification does NOT mean that there has been a terrorist action. Instead, it means that a recognized event is properly turned into a threat notification, according to the expectation of an experienced user (typically an operational user) .

The threat detection activity is carry out by Micro-environments. A Micro-environments is a software component that interprets events coming from a limited zone in order to identify potential threats. For the Micro-environments implementation it is used an HMM approach.

5.1. Hidden Markov Model

Hidden Markov Model (HMM) is a statistical Markov model in which the system being modeled is assumed to be a Markov process with unobserved (hidden) states. One of the bases of HMM is the Markov Chains. A Markov Chain [Norris, 1998] is a stochastic process with the Markov property on a finite or countable state space. The term Markov Chain refers to the sequence (or chain) of states such a process moves through. Usually a Markov chain is defined for a discrete set of times. The Markov property states that the conditional probability distribution for the system at the next step depends only on the current state of the system, and not additionally on the state of the system at previous steps.

In Markov Chain the changes of state of the system are called transitions, and the probabilities associated with various state-changes are called transition probabilities. The process is characterized by a state space, a transition matrix describing the probabilities of particular transitions and an initial state or initial distribution across the state space.

In order to define a Markov Chain model, the following three elements have to be defined:

1. The N states of the Model, defined by $S = \{S_1, \dots, S_N\}$
2. The State transition probability distribution $= \{a_{ij}\}$, where a_{ij} is the probability that the state at time $t + 1$ is S_j, is given when the state at time t is S_i. The structure of this stochastic matrix defines the connection structure of the model. If a coefficient a_{ij} is zero, it will remain zero even through the training process, so there will never be a transition from state S_i to S_j.

$$a_{ij} = p(q_{t+1} = j \mid q_t = i), 1 \leq i, j \leq N$$

   Where q_t denotes the current state. The transition probabilities should satisfy the normal stochastic constraints, $a_{ij} \geq 0, 1 \leq i, j \leq N$ and $\sum_{j=1}^{N} a_{ij} = 1$.

3. The initial state distribution $\pi = \{\pi_i\}$, where π_i is the probability that the model is in state S_i at the time $t = 0$ with $\pi_i = p(q_1 = i)$ and $1 \leq i \leq N$.

5.2. Description of HMMs approach

A Hidden Markov Model (HMM) is a finite stochastic automaton [Dymarski, 2011] that summarizes a kind of double stochastic process:

- The first stochastic process is a finite set of (hidden) states, each associated with a multidimensional probability distribution. The transitions between states are statistically organized by a set of probabilities (transition probabilities). In the thesis's context two hidden states (Threat and Not Threat) model whether a threat is present in a Micro-environment; more precisely, the probability of the Threat state models is the probability of a threat in the Micro-environment.

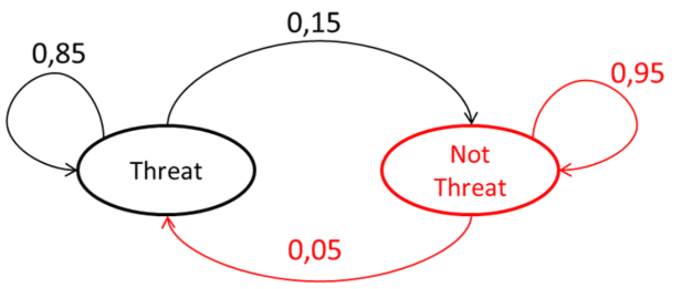

Figure 12 HMM transition states.

- The second stochastic process models the probability that an event is observed in each state (observation probability). In this case observable events represent abnormal situations observed by sensors (including low-level processing algorithms and human beings). Abnormal situations are classified according to several levels of relevance; for example NotRelevant (NR), Low (L), Medium (M), High (H), AlarmCeased (AC).

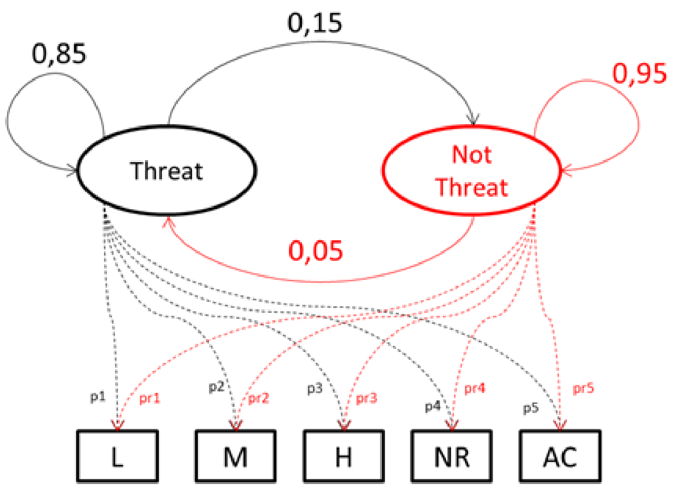

Figure 13 HMM observation probability.

The model evolves at discrete times. At each step the new probability of the hidden states is computed according to the transition probabilities and tuned according to what has been observed and to the observation probabilities in each state. The result is that the new probabilities of the states subsume in a synthetic way the previous history of observations.

In general, the “natural” evolution of the model in absence of relevant abnormal situations leads to lower probabilities of the Threat state, whereas the observation of abnormal situations raises it (see also the discussion about alert levels in section 5.5).

Whenever the probability of the Threat state overcomes a threshold, a threat notification is delivered in real time to operational users and is stored for further processing (Medium and Long-term TENSOR activities or users' consultations).

The model is trained by capturing both the states (i.e., the presence of “real” threats) and the corresponding observations. This can be done by exploiting either real data or data simulated. In general, an experienced user working at the operational level should be in charge of rating the threat notifications i.e., to recognize false positives and (possibly) negatives.

5.3. Mathematical definition

Each Hidden Markov Model is defined by states, state probabilities, transition probabilities, emission probabilities and initial probabilities.

In order to define an HMM completely, the following five Elements have to be defined:

1. The N states of the Model, defined by $S = \{S_1, \dots, S_N\}$
2. The M observation symbols per state $V = \{V_1, \dots, V_M\}$
3. The State transition probability distribution $= \{a_{ij}\}$, where a_{ij} is the probability that the state at time $t + 1$ is S_j, is given when the state at time t is S_i. The structure of this stochastic matrix defines the connection structure of the model. If a coefficient a_{ij} is zero, it will remain zero even through the training process, so there will never be a transition from state S_i to S_j.

   $$a_{ij} = p(q_{t+1} = j \mid q_t = i), 1 \le i, j \le N$$

   Where q_t denotes the current state. The transition probabilities should satisfy the normal stochastic constraints, $a_{ij} \ge 0, 1 \le i, j \le N$ and $\sum_{j=1}^{N} a_{ij} = 1$.
4. The Observation symbol probability distribution in each state, $B = \{b_j(k)\}$ where $b_j(k)$ is the probability that symbol v_k is emitted in state S_j.

   $$b_j(k) = p(o_t = v_k \mid q_t = j), \qquad 1 \le j \le N, \qquad 1 \le k \le M$$

   where v_k denotes the k^{th} observation symbol in the alphabet, and o_t the current parameter vector. The following stochastic constraints must be satisfied:

   $$b_j(k) \ge 0, \qquad 1 \le j \le N, \qquad 1 \le k \le M \text{ and } \sum_{k=1}^{M} b_j(k) = 1$$

5. The HMM is the initial state distribution $\pi = \{\pi_i\}$, where π_i is the probability that the model is in state S_i at the time $t = 0$ with $\pi_i = p(q_1 = i)$ and $1 \le i \le N$.

5.4. Assumption

We will make two Markov assumptions:

1. The limited horizon assumption is that the probability of being in a state at time t depends only on the state at time $t-1$. The intuition underlying this assumption is that the state at time t represent “enough” summary of the past to reasonably predict the future.
2. The stationary process assumption is that the conditional distribution over next state given current state does not change over time.

Additional assumptions in TENSOR context:

3. Micro-environments are characterized by an alert level, which is set by MTR or expert users according to medium/long term forecasts. We define three alert levels (Low, Medium, High). Therefore there are three different HMMs; which model is active depends on the criticality (summarized by the alert level) of the Micro-environment.
4. The three models differ for the lowering speed of the probability of having a threat in absence of relevant abnormal situations (on Low criticality the probability of Threat decreases faster than on High criticality where that probability remains constant).

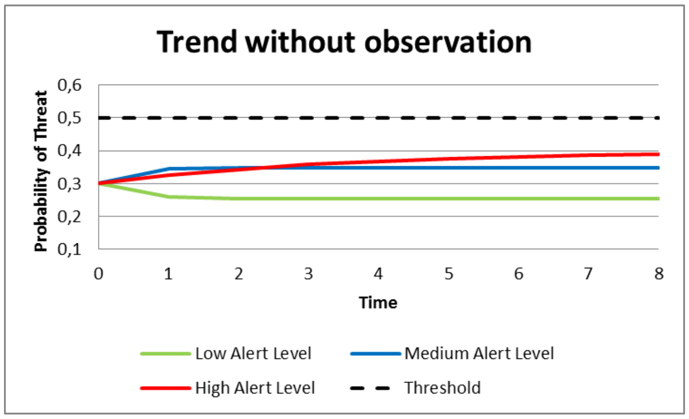

Figure 14 Trend of threat probability without observation.

5. The three models differ for the effect of the observations (on High criticality the observation of an abnormal situation raises the probability of Threat faster). In particular:
    a. On High alert level a single low relevance anomaly is sufficient to overtake Threat threshold (Figure 15) while on Low alert level the probability of Threat increases but does not reach the threshold (Figure 16).
    b. In the case of multiple low relevance anomalies, models behave in the same way. On High alert level the probability of Threat remains above the threshold (Figure 17) while on Low alert level it would take a large number of low relevance anomalies to reach the threshold (Figure 18).

6. When the alert level changes, the last state of the current model becomes the initial state of the new model (Figure 19).

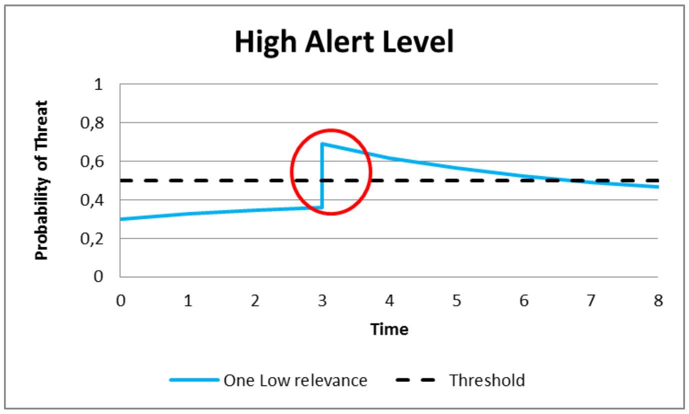

Figure 15 High Alert level with one Low relevance observation.

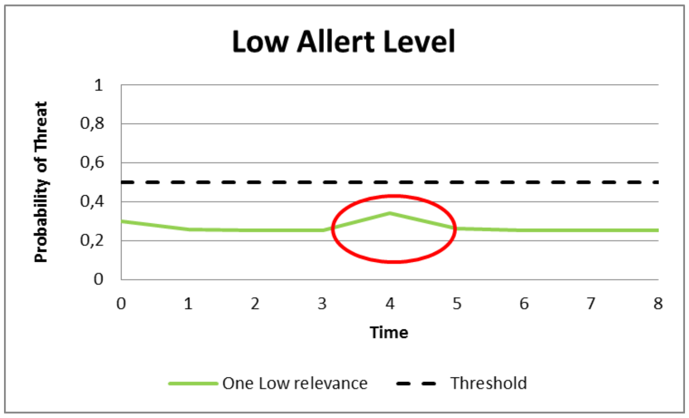

Figure 16 High Alert level with one Low relevance observation.

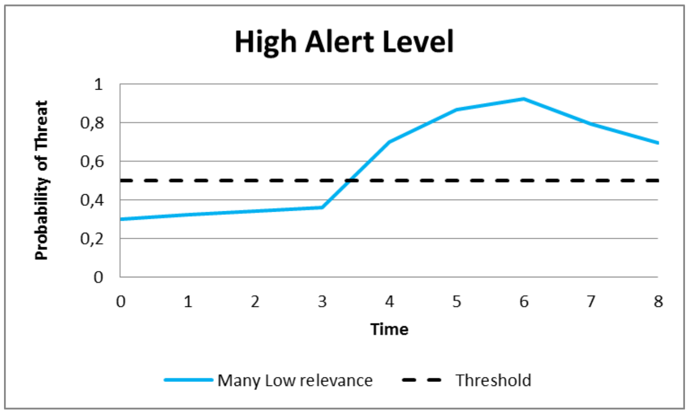

Figure 17 High Alert level with many Low relevance observations.

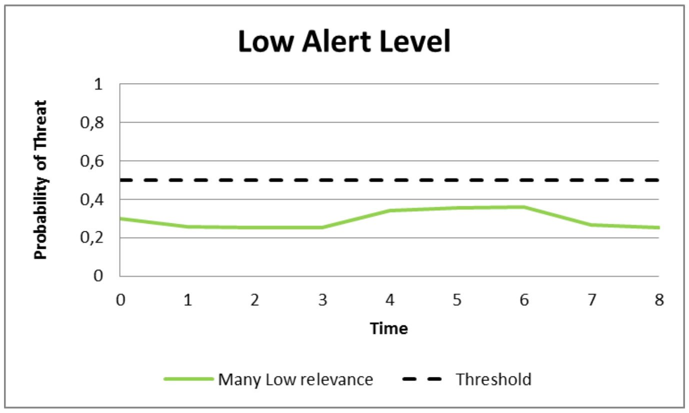

Figure 18 Low Alert level with many Low relevance observations.

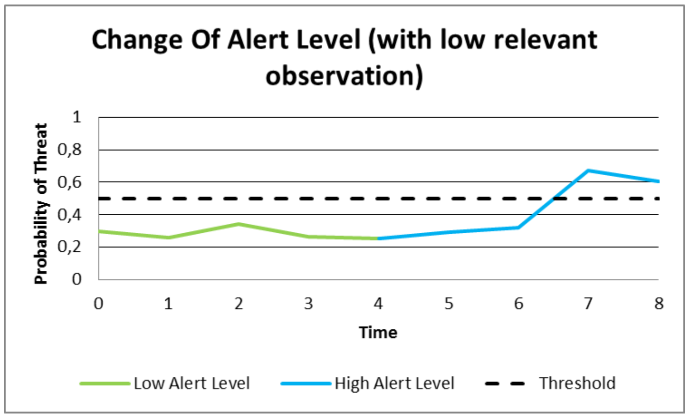

Figure 19 Change of alert level: from Low to High alert level.

5.5. Alert levels

The behavior of a Micro-environment is basically "context-free", as the HMM micro-model relies on its local history only. Though this approach provides a sound basis for a highly modular and extensible system, its major drawback is that it does not take into account the overall state of the scenario. Therefore MTR is in charge of tuning the alert level of individual Micro-environments, so that they behave differently according to their alert level.

The tuning of the alert level of individual Micro-environments is performed by MTR on a medium-term time scale by relying both on overall forecasts and on hints provided by strategic human decision makers.

The alert level is a sort of compromise between false positive and false negative threats detection. When the monitored zone is in a normal (or quit) situation (i.e. when the probability of a threat is very low? "Low alert level") a high percentage of false positive isn't accepted but there is a greater tolerance for false negative (in low alert level some symptoms of potential threat can be "neglected"). In a "Low alert level" the gap from false positive and false negative is unbalanced in favor of false negative.

If the alert level changes from Low to Medium this gap thins. In a “Medium alert level” situation, there is a higher probability of threats; thus a higher percentage of false positive and a lower percentage of false negative is accepted.

Accordingly, during a period of “High alert level” (i.e. a situation where there is almost the certainty of a threat) the gap between false negative and false positive changes in favor of false positive. In this situation the tolerated percentage of false positive is very high but there aren’t tolerances for false negative occurrences.

To get this behavior each Micro-environment includes three HMMs, one of which is selected according to the alert level. In this way it is possible to adapt the behavior of a Micro-environment when a particular interest for an environment by terrorist is forecasted (medium term activities).

A change of alert level is performed by MTR activities when:

- There is one or more potential threats symptoms in a physical monitored environment or into the neighbor environments (STR threats evaluation)
- There is a new forecast about particular interest for an environment by terrorists (LTR long-term activities)
- A strategic user decides to change one or more alert levels basis on his experience or reasoning

5.6. Micro-environments

A Micro-environment is a software component, which runs in the TENSOR architectural framework presented in section 4.1.3. It models a target (for example a building, a metro station, a limited set of streets…), which can be monitored by a set of virtual sensors, be they devices or persons. A Micro-environment:

- Gets events (abnormal situations) from virtual sensors
- Recognizes the relevance of the events according to a static, Micro-environment specific mapping
- Adjusts its threat state according to its Hidden Markov (micro) Model
- Notifies threats to operational user interface and stores it into the database.

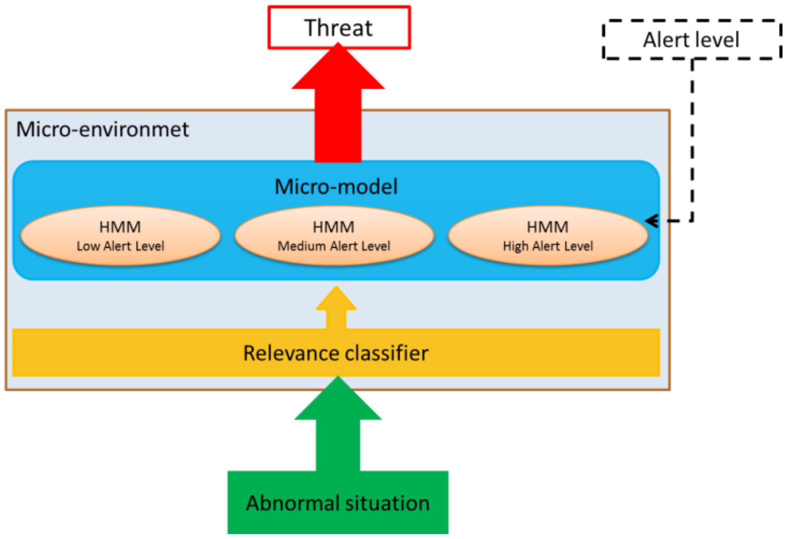

Figure 20 Micro-environment structure.

A Micro-environment first classifies an abnormal situation according to its relevance. This is up to an inner layer, which defines the relevance of the abnormal situation in the specific context of the Micro-environment (for example, a parked car in a parking lot may be of low or no relevance, whereas a car parking in front of the Ministry may be of medium or high relevance). The mapping from abnormal situations to relevance levels is static and context-specific. It depends on the specific Micro-environment, whereas it does not depend on the current threat state nor on the criticality of the Micro-environment.

Then a Micro-environment adjusts its threat state according to its Hidden Markov (micro) Model and possibly generated a threat.

Micro-environments are characterized by an alert level, which is tuned by higher system layers (Medium Term Reasoning or expert users) according to medium term forecasts. The alert level affects the behavior of the micro-models in order to dynamically get a reasonable compromise between false negatives and information overflow.

A Micro-environment component receives as input:

- Events (from virtual sensors)
- Alert level (from MTR)

If the Micro-environment states that there is a dangerous situation it returns as output a threat notification.

5.6.1. **Micro-models**

A micro-model summarizes the threat state of a Micro-environment by relying on a Hidden Markov Model.

Each micro-model is composed by three different HMMs, one for each alert levels (high, medium and low). At a specific time instant, only the HMM associated with the current alert level is used to detect potential threat situation.

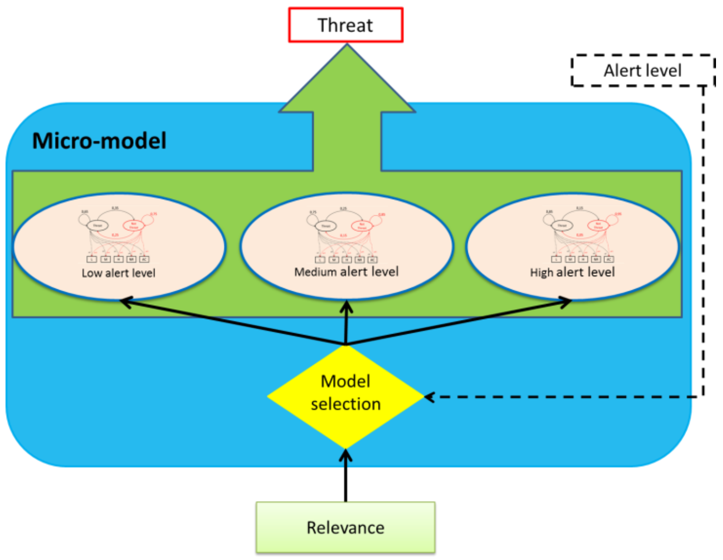

Figure 21 Micro-model structure.

As seen before, the inputs of micro-models are values of relevance associated to incoming events. Before the threats detection activity, each Micro-environment classifies the events according to the specific static mapping from events to relevance that depends on the features of each physical zone and is established by expert users (typically operational or tactical users). The introduction of this classification stage allows:

- Clustering events into macro-categories according to their relevance
- Limit the size of the HMM observation matrix
- Decrease the training time and complexity of the micro-models.

5.6.2. **Training the micro-models**

This activity is focus on the definition and training of the micro-models. This long-term activity (performed by LTR) produces the micro-models, which are viewed as “nearly-static” in the context of the Micro-environments performing short-term activities.

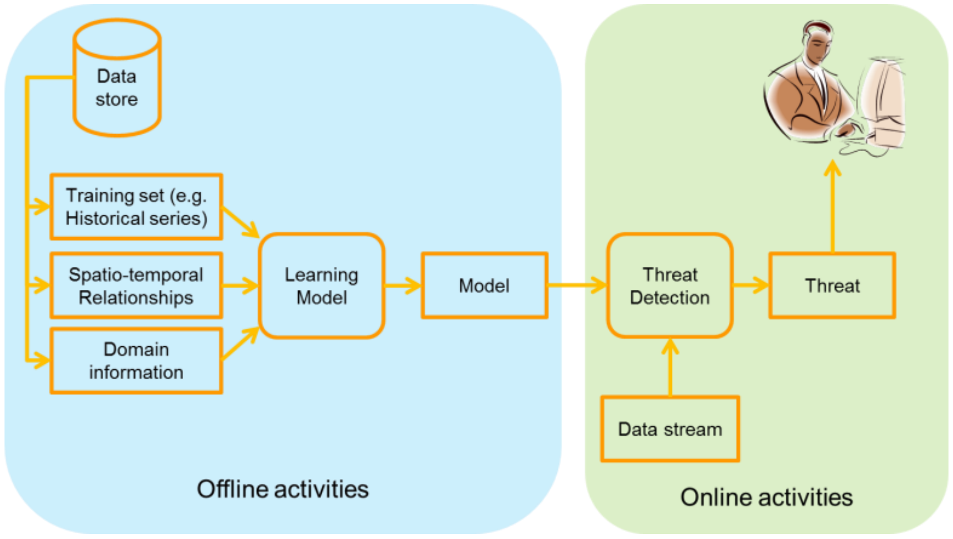

Figure 22 Micro-models training.

Note that this activity is implemented by the component that does not run in the STR module. They interact with STR components via database only. This is coherent with the fact that real-time threat recognition (Short-term activities) and offline micro-model generation (Long-term activities) are performed at different time scales.

5.6.3. **Exploiting historical series**

The training micro-models activity requires information about:

- Historical series about events including for each event:
    - Space contextualization (each historical series includes events occurred in a specific physical zone)
    - Time stamp (instant at which the event has been identified by a virtual sensor)
    - Event description
    - The current alert level of the Micro-environment
- Historical series about potential threat situations including:
    - Space contextualization
    - Time stamp
    - A reference to the event (or events) that generated the potential threat situation
- The mapping relation from events and their value of relevance (for each monitored physical zone).

From that information it is possible to build, for each zones and for each alert level, several historical series composed by pairs {Threat State, Abnormal situation}, where Abnormal situation is the value of relevance associated with each event and the Threat state represent if that Event represents a significant symptom of a possible attack. After this “build” phase of historical series it is possible training each micro-model.

5.7. Gaming data

A serious set-back to the development and operation of appropriate reasoning and analytics techniques for anticipating terroristic events is the lack of ground-truth datasets, which could be used to train the respective reasoners. Typically, LEAs do not possess large sequences of events that are associated with terrorist attacks, since their databases are usually limited to the set of events that have been observed before the occurrence of past attacks.

One approach to simulating and generating such data sets employs serious games [Bruzzone et al., 2009], notably games that are used to train security forces into assessing and responding to a similar situation. However, these games are appropriate not only for training officers in effectively engaging in terroristic situations, but also for:

- Developing expert system rules that can be used to raise alarms on the risk of terrorist attacks
- Generating data sequences that will be used to train machine learning algorithms.

In this thesis, a serious game is used for the generation of datasets that were used for the micro-models training activity. The game involves two parties (i.e., the LEA and the attackers) which have conflicting goals. In particular, the attacker attempts to complete a number of prerequisite steps prior to launching an attack, notably the steps identified in [Bennet, 2007], which are conveniently called terroristic indicators. On the other hand, the LEA attempts to identify the actions of the terrorists (early on), thereby preventing an attack.

The PROACTIVE game

The PROACTIVE game [Sormani et al., 2016] is used in order to generate training data sets for the Micro-environments, notably training data sets that comply with the event detection dictionary (section 4.1.1). It is a turn based game, typically played by two players (namely the LEA and the terrorists) and it is implemented as an Excel WorkBook (with multiple sheets). The game is divided in time slots, in which each player makes a move. The LEA and the terrorists use different sheets of the WorkBook ("Police Moves" and "Terrorist Moves") to enter their moves so that they can't see each other's moves.

The players (typically the LEA), before play a round, they have to specify:

- The set of monitored zones (M)
- The set of actions that can carry out a terrorist (A)
- The set of resources that the police use to monitor the zones, which can be:
    - Devices (D), like cameras
    - Human patrols (P)

In the PROACTIVE game, it is always assumed that the LEA does not have the resources needed to cover all the monitored zones (i.e. $|M| > |D| + |P|$). There is no restriction to moving cameras around zones, but in this work thesis, in order to simulate a more realistic scenario, the zone of each camera has been fixed, while human patrols can move between zones (note that a zone can have both camera and human patrol assigned to it).

The goal of the game is for the terrorists to complete an attack against a specific zone and for the LEA to prevent the attack. The terrorists need to:

- Observe a specific zone 4 times (either in-vehicle or on-foot)
- Acquire information on the target (i.e. the zone)
- Test the security of the checkpoint and
- Execute the attack.

Each one of the previous steps can be executed only if all previous steps have been already executed. However, steps for different zones might be performed in an alternating way (but not in the same timeslots). For each one of the listed steps, the terrorist actions might generate events (belonging to the event detection dictionary) that are detected by the police. The generation of an event from an action is a random process which depends on whether police force is assigned to guard and/or a camera to survey the specific zone at the time slot of the terrorists' action. In Table 6 some examples are reported.

Table 6 Probability of actions generating events.

Terrorist action	**Detected event**	**Probability of detection by the police**	**Probability of detection by camera**
On-foot surveillance	Person observing building	1	0,5
Test security checkpoint	Security checkpoint alarm	1	0,4
Place bomb (execute the attack)	Suspicious unattended object spotted	0,9	0,6

So, the LEA sees the moves of a terrorist in specific zones in cases where they have deployed cameras or human patrols in those locations and if the device or the patrol is able to identify the event. Similar to the previous case, the terrorist sees whether their actions have been perceived by the LEA according to a configurable probability.

The LEAs detect and prevent the terrorist attack (and win the game) if two events of any type are generated for the same location (this number can be set by the players during the configuration phase). The terrorists win if the 7 steps are completed for a single location and at most one event has been generated for each location (in this case it is assumed that the police have not detected the terrorists).

Validation scenario

The output of playing the PROACTIVE game is sequences of events containing the following information: a time stamp, the sensor (human or device) which detects the event, the zone in which the event was detected and who (terrorist or normal people) carried out the action that triggered the event. All this information is used in order to train the different Micro-environments.

The validation scenario (Figure 23) included three different zones: the Ministry of Finance, the square and the church. The Ministry of Finance and the square are close to each other while the church is further away.

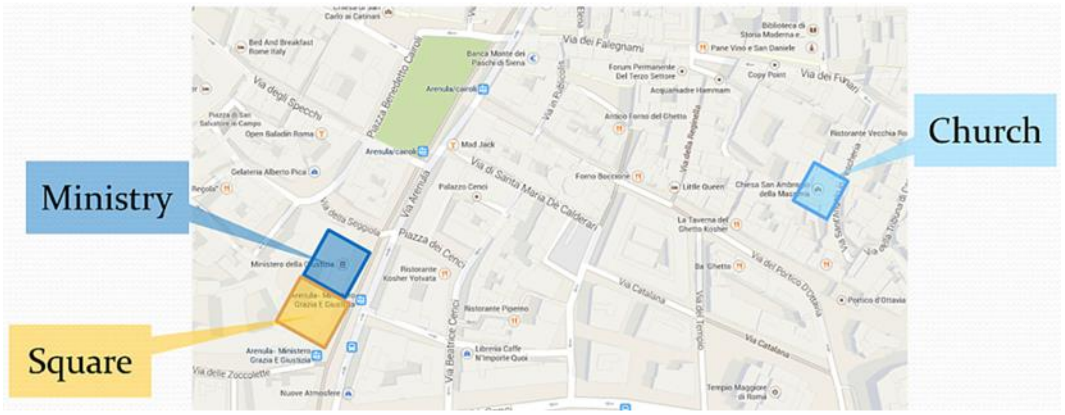

Figure 23 Validation Scenario Map.

The scenario includes one camera for each zone and one human sensor (policeman), that can move across areas according to the gaming. Gaming sessions were played for each zone and different alert levels (sensitivity of situation). Generated sequences of events span over a predefined time of 2 days horizon (e.g., 04/03/2015 and 05/03/2015) and include terrorist moves observed by either a police patrol (i.e., human sensor) or detected by a camera. They also include around 60% of events that are not generated as gaming moves but simulate the behavior of normal people. The STR modules were trained using these sequences. Table 7 reports some Short Term Events (first column of Table 7) that can be detected from sensors and the output of Micro-environment analysis (third column of Table 7). As it is possible to see, the Micro-environment output changes according to its associated alert level (second column of Table 7).

Table 7 Symbolic Event Example.

Symbolic Event	**Micro-environment Alert Level of the**	**Micro-environment output**
Person Observing Building	Low	No threat notification.
	Medium	No threat notification for a single event. Threat notification for frequently repeated events.
	High	Threat notification even for single event.
People Not Fitting In Environment	Low	No threat notification for a single event. Threat notification for frequently repeated events.
	Medium	Threat notification even for single event.
	High	Threat notification even for single event.
Unusual Movement Of Vehicle Close To Target	Low	Threat notification even for single event.
	Medium	Threat notification even for single event.
	High	Threat notification even for single event.

In order to evaluate the training phase of Micro-environments, additionally testing sequences were generated for all three physical environments and for different alert levels and span over a period of

4 hours from 9AM to 12PM the following day 06/03/2015. The average length of these sequences is 110 events. In this testing scenario where only STR was used without recommendations/update of alert levels by the MTR and LTR module, we asked domain expert users to check the notification output of Micro-environments. In particular they were asked to evaluate for each event if the Micro-environment threat notification is right (i.e., in this particular time instance, with this specific alert level, this event is rightly/wrongly labeled as a potential symptom of a threat situation) or if the missing threat notification is right (i.e., this event is rightly/wrongly labeled as not a potential symptom of a threat situation). The information provided to expert users was:

1. The event sequence generated with gaming
2. The Micro-environment alert level
3. The output of Micro-environment (threat/no threat).

In order to evaluate the Micro-environment threat detection performance, we decided to use traditional machine learning metrics like:

- Precision: the ratio of the number of relevant threat notifications to the total number of irrelevant and relevant threat notifications.
- Recall: the ratio of the number of relevant threat notifications to the total number of relevant threat notifications in the considered set.
- F-Measure: the harmonic mean of precision and recall.

Table 8 Traditional evaluation metric.

	Alert level		
Metric	**Low**	**Medium**	**High**
Precision	0,75	0,72	0,88
Recall	0,54	0,58	0,80
F-Measure	0,63	0,65	0,84

The discrete performance evaluation obtained calculating the traditional metrics (Table 8) has led us to identify some different metrics that can be inferred from the user's expectations. So, we asked domain experts users what they expected from the threat detection modules (i.e., Micro-environments) and in particular how the Micro-environments should be able to change their threat detection strategy according to the selected alert level. As expected, different users provided different answers, but all the users have expressed the need to have a threat detection module that was able to use a more severe strategy when the alert level is set to "High" and a soft strategy when the alert level is "Low". In other words, they prefer to have a lot of threat notifications, even if the event that trigger the threat notification may not be a potential symptom of a threat situation, when they set the alert level to "High" and they accept to lose some notifications (i.e., events that may be symptoms of a threat situation are not notified) when they set the alert level to "Low". According to this concept, the evaluation metrics selected for the evaluation of the STR module are:

- False positive rate = $\sum$ False positive / $\sum$ Condition negative
- False negative rate = $\sum$ False negative / $\sum$ Condition positive

Where

- ∑ Condition positive (negative) represents the total number of the events that expert users recognize as potential (not potential) symptoms of threat situations.
- False positive represents the total number of threat notifications related to events that expert users recognize as not potential symptoms of threat situations.
- ∑ False negative represents the total number of missing threat notifications related to events that expert users recognize as potential symptoms of threat situations.

Table 9 User's evaluation metrics.

	Alert level		
Metric	**Low**	**Medium**	**High**
False positive rate	4,44%	15,15%	16,67%
False negative rate	45,45%	41,30%	20,00%

The results reported in Table 9 reflect the user expectation since it is evident that the False positive rate increases with increased alert level while the False negative rate decreases when the alert level decreases.

6. Long Term Reasoning layer

As introduced before, the Long Term Reasoning has the aim to provide help at strategic users. In Figure 24 (red lines) is shown the two main activities carry out by LTR, the alert level prediction and the modules update activity.

The main goal of the LTR is to identify the alert levels of each type of physical environments through the analysis of historical data (for example external data sources containing previous terrorist behaviors and attacks) and observed threats (events history).

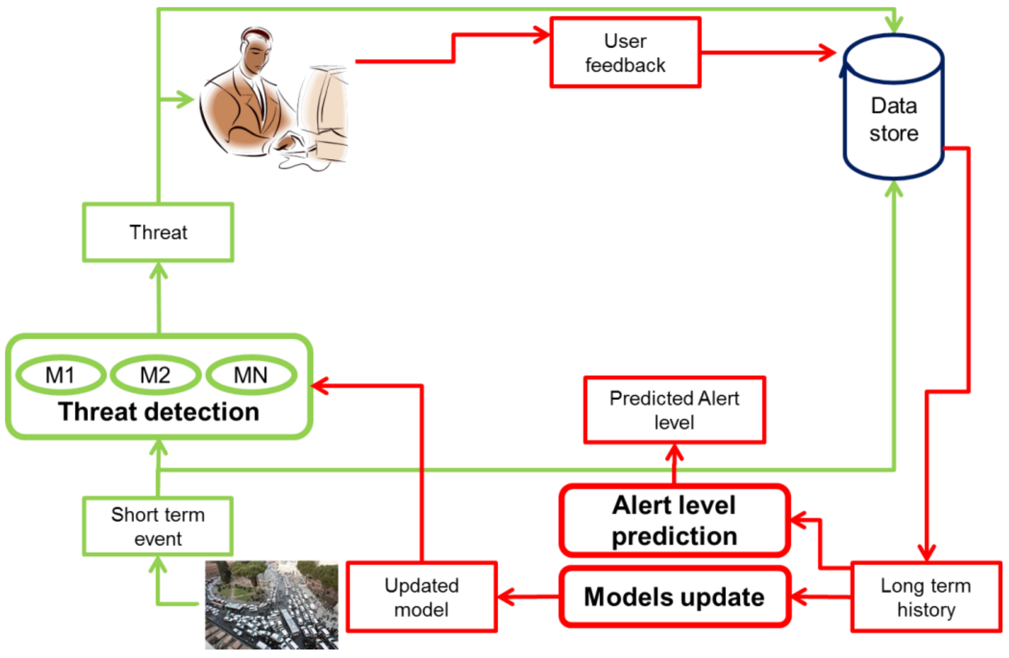

Figure 24 Long Term Reasoning (red lines).

The alert level prediction activity provides information about alert levels for specific types of Micro-environments that correspond to types of physical zones (e.g., public buildings, metro stations, and so on). For this activity, the LTR uses information provided by external sources (e.g., external historical datasets), like intelligence scenarios and/or scheduled activities (e.g., planned high-profile events like “visit of a minister”).

The LTR also performs the Micro-environments update activity using the data stream provided by sensors (i.e. detected events), the detected threats provided by Micro-environments and the operational user feedback collecting during the interaction with the system.

Pre-processing steps

Data pre-processing is one of the most critical steps in the data mining process which deals with the preparation and transformation of initial dataset. Also in this case we apply different pre-processing steps to the GTD (section 2.5.3) dataset starting from the feature selection step to the missing values replacement and feature binarization.

Following a brief description of each step:

- Feature selection: as described into the [GTD Codebook 2015] the initial GTD dataset consists of 124 features. During the feature selection step we have mapped the GTD feature space into the validation scenario ones (see section 5.7). In this way we have obtained a new dataset configuration containing features describing the actions (the activities carried out during an attack) and the context of the attack. In particular we have 36 features describing the actions and 22 describing the context. Finally all the features has been binarized (e.g. assume 0 or 1 value).
- Missing values analysis: One of the other main steps to apply before analyzing a dataset, is the so called "missing values replacement"; this step is very important since the presence of missing data can reduce the representativeness of the sample and can therefore distort inferences about the population.

6.1. Mapping between GTD data and event dictionary

As stated previously, in order to use the GTD database as data sources for the Long Term Reasoning component, it is necessary to map GTD feature space information and the validation scenario ones.

Two different kinds of mapping have been applied:

- Physical environment mapping: mapping between GTD "target type" feature and the validation scenario physical environment. For simplicity we note that the GTD target type feature captures the general type of the attack and consists of 22 different categories.
- Event mapping: mapping between action features defined in GTD and events that can be provided by intelligence or LEAs. These events are called "long term events" because they are not processed by Micro-environments but they are store into the database typically by tactical/strategic user's. However LTR takes also in consideration "short term events" provided by sensors and the threat notification provided by Micro-environments.

In the following tables we reported the two mappings explained above.

Table 10 Physical environment mapping.

GTD Target Type	Validation Physical Environment
Government	Ministry of Finance
Tourists	Square
Religious figures/institutions	Church

Table 11 GTD action features mapping.

GTD Action Features	GTD Action Features Values	Validation Scenario Events
Attacktype	Assassination	Person killed
	Armed assault	Pistol found, firearm found
	Bombing/explosion	Bomb found
	Hijacking	Bus hijacked
	Hostage taking (barricade incident)	Person taken hostage
	Hostage taking (kidnapping)	Kidnapped person
	Facility / infrastructure attack	Sabotage security system, test security system
Weaptype	Handgun	Handgun found
	Grenade (not RPGs)	Grenade found
	Mine	Mine found
	Projectile (rockets, mortars)	Projectile found
	Vehicle	Suspicious vehicle
	Arson/Fire	Fire detected
	Knife or Other Sharp Object	Knife found
	Dynamite/TNT	Dynamite found
	Sticky Bomb	Bomb identified
Hostkidoutcome	Hostage(s) killed (not during rescue attempt)	Hostage killed
Claimmode	Letter	Letter
	Call (post-incident)	Call (post-incident)
	Call (pre-incident)	Call (pre-incident)
	E-mail	E-mail
	Note left at scene	Note left at scene
	Video	Video
	Posted to website, blog, etc.	Posted to website, blog, etc.
	Personal claim	Personal claim
Ishostkid	The victims were taken hostage or kidnapped	Person kidnapped
Ransom	The incident involved a demand of ransom.	Ransom request

6.2. Prediction activity

Taking into account all the considerations reported in the previous subsection, a clustering approach is used to model “sets of actions” that map to the validation scenario symbolic events; furthermore we analyze their association with contexts which map to types of validation physical environments. The goal of LTR is to capture commonalities across different terroristic actions and contexts in order to predict alert levels for different types of physical environments based on event history.

In order to achieve this we use spectral clustering as it very often outperforms traditional clustering algorithms (e.g. k-means algorithm, k-nearest neighbors). The main reason for better performance lies in the fact that spectral clustering doesn’t make any assumptions regarding the shape of it clusters and can therefore model different cluster shapes that need not be necessarily convex as in case of k-means. Furthermore, spectral clustering can be efficiently implemented even for large datasets in case the similarity matrix is sparse. Still it is a very sensitive approach regarding choices of parameters and if carefully configured can give extremely good results. Therefore in this section we describe our approach and perform detailed parameter and performance analyses for the specific terrorist domain. An additional reason for selecting a clustering approach is its ability to deal in an unsupervised manner with large sets of history events.

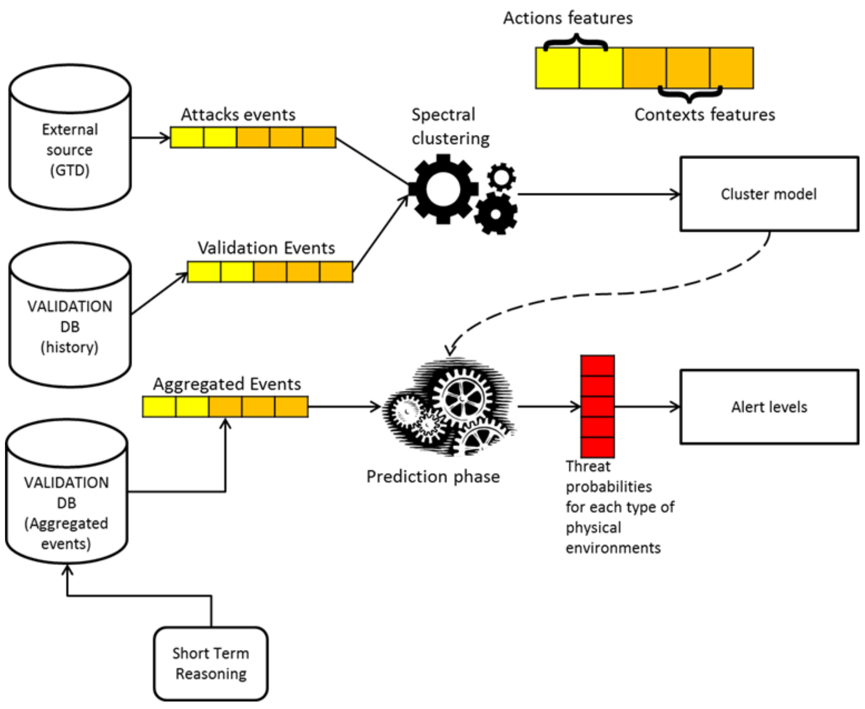

Figure 25 Long Term Reasoning module.

The approach (depicted in Figure 25) allows us to group instances (either validations histories or terrorist attacks from GTD) from the selected dataset into clusters that ideally capture models of similar threat events (validations history) and/or of attacks (in case of GTD dataset). The instance are represented as multi-dimensional feature vectors comprised of action features which are mapped to validation events and of context features which are mapped to types of physical environments in the validation scenario (see section 5.7 for more details):

$$f_i = [f_a^i, f_c^i]$$

where f_i is the feature vector for the instance , $i = \overline{1, N}$ is the number of instances in the dataset, f_a^i and f_c^i are the feature vectors in the action and context subspaces.

The action feature space is used as bases for building the cluster models and capturing the underlying similarity. Furthermore as for each instance the set of action features is linked to a corresponding set of context features, we capture the distribution of different physical environment context features across the clustered action feature space. We assume that in this way we can model the correlation between combinations of actions (validation scenario events) and corresponding physical validation scenario environments.

Through building these models we aim at capturing the “typical behavior” of a terrorist attack that can potentially lead to a threat and the related physical environment.

In the analyses and verification stage described in the remaining of the section, we measure the ability of this approach to correctly predict the types of physical environments related to the particular combination of actions.

In the operational run-time stage when new events are introduced in the system enabling histories of threat/no threat events to be used. The reasoning phase of this module will take as an input aggregated event capturing the short-term histories of threats and validation events across a time window of analyses (~1h) and position it in the closest cluster defined in the action feature space. Furthermore we can now take into account the previously captured associations of different context features (mapping to physical environments) across action clusters and predicts the amount of presence of each type of validation physical environments in the identified cluster. These numbers together with the presence of threat events enable us to infer the probability of a threat event happening in each type of physical environment leading to corresponding alert level values.

Briefly, the activities of LTR can be summarized as follows and reported in more detail in subsections 6.2.1 and 6.2.2:

1. Building models representing different groups of instances (attacks from GTD or validation events from history) using the action feature space and calculating the corresponding associations of context features over the clusters
2. Generating aggregated events representation that captures the state of the STR component during the last time window of analysis (~1h).
3. Identifying the cluster that is closest to the aggregated events in the action feature space and estimating the relevance of threat for each type of physical environment for the identified cluster hence their criticality values
4. Passing the predicted criticality levels (PCLs), to the MTR module

As depicted in Figure 25, the LTR module can be divided into two different phases: the model generator phase that learns the cluster models and the long-term prediction phase that using these models to infer the criticality levels of each type of physical environment.

6.2.1. **Spectral clustering vs k-means**

For completeness we are also analyzing performances of our implemented spectral clustering approaches in comparison to those obtained by K-means clustering approach.

We show only this comparison, however we have performed additional tests with other state of the art clustering algorithms like farthest-first, partitioning around medoids (PAM).

In this case, we report the performances of both algorithms taking into account the Silhouette and Calinski Harabasz indexes and varying the number of clusters.

Different configurations of both spectral clustering algorithm and k-means algorithm have been tested, Figure 26 and Figure 27 depict the results obtained with the best configuration of the Spectral Clustering algorithm (using hamming distance and L =2) and the K-means one.

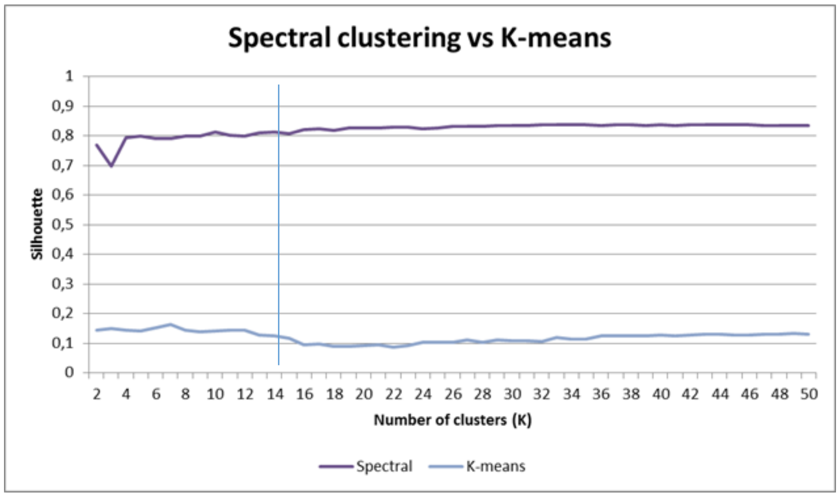

Figure 26 Spectral clustering vs K-means (Silhouette index).

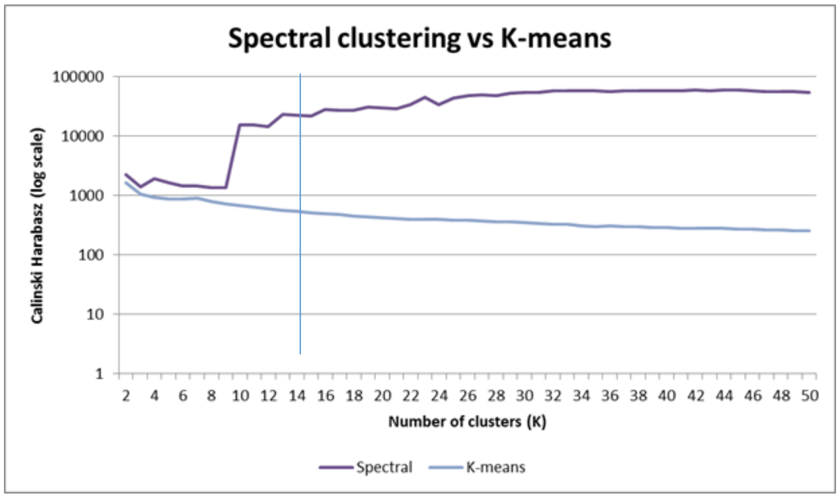

Figure 27 Spectral clustering vs K-means (Calinski Harabasz index).

From the above picture, we can conclude that the spectral clustering outperforms the k-means approach. We take this as a confirmation for the choice of spectral clustering as our approach. Basically classical clustering approaches achieve lower performance than the worst case obtained with spectral clustering approach.

6.2.2. **Experimental settings and performance evaluation**

In the analyses and verification stage described in the follow-up we measure the ability of our approach to correctly predict types of physical environments related to a particular combination of actions. For these experiments we used a subset of the GTD dataset with 10,000 instances. We select a subset in order to verify the performance of our approach.

The set of $N = 10{,}000$ instances is randomly selected from the database and we run our experiments with 10-fold cross validation. In every iteration of the learning algorithm 9 folds of data were used to learn the model, and 1 fold to make the predictions. We experimented with different parameter configurations in respect of distance metrics, number of eigenvalues and cluster numbers. As discussed in section 6.2 for GTD we have categorical features describing information regarding actions as well as categorical features describing context. In the binarized feature space we end up with 278 features for the action feature space and 22 features the context feature space.

For each feature $f_a^i, i = \overline{1,278}$ of the action feature space and each feature of the context feature space $f_c^i, i = \overline{1,22}$ we calculate T_a^i and T_c^i, the percentages of instances that contain that particular feature across the database with N instances.

We then perform spectral clustering using only the action feature space and obtain K clusters. Furthermore for each cluster k , where $k = \overline{1,K}$ we calculate a centroid c^k in the following manner:

- We calculate the presence of each feature from both the action and context feature spaces in the cluster; that is T_a^k and T_c^k, the percentages of instances of the cluster k that contain that particular feature across the whole cluster with N_k instances.
- The cluster centroid $c^k = [c_a^k, c_c^k]$ vector contains two sub-vectors representing a centroid c_a^k in the action feature space in which the clustering into k clusters was performed, and a centroid in the context features space c_c^k by taking into account the context features of the member instances of cluster k.
- We generate values of the centroid $c_a^k = [c_{a,i}^k]_{i=\overline{1,278}}$ in the following manner, for each feature f_a^i of the action feature space if the value $T_a^k \geq T_a^i$ then $c_{a,i}^k = 1$, otherwise it's 0.
- We perform exactly the same steps for the centroid in the context space $c_c^k = [c_{c,i}^k]_{i=\overline{1,22}}$, if the value $T_c^k \geq T_c^i$ then $c_{c,i}^k = 1$, otherwise it's 0.
- When a new instance is used in the testing phase, we calculate the distance of the new example from each centroid in the action feature space c_a^k, $k = \overline{1,K}$ using distance metric M. The closest centroid indicates the cluster into which we classify our new instance. In this manner the values of the centroid in the context feature space c_c^k are used as predicted labels for each feature of the context space (where each context feature maps to a type of a validation physical environment). In this way our problem can be viewed as a multi-class classification scenario.
- F-measure and accuracy are calculated in a macro fashion that is, across all classes for each new instance, and then averaged in a 10-fold cross-validation scenario.

The above mentioned steps were repeated for different configurations by combining following parameters: distance metrics M (Hamming, Jaccardbin and Cosine), L of eigenvector L=2, number K of clusters (from 2 to 50).

For completeness, a brief definition of the two measures is reported:

- Accuracy:

$$\text{Accuracy} = \frac{\text{tp} + \text{tn}}{\text{tp} + \text{tn} + \text{fp} + \text{fn}}$$

Where: tp= true-positive, tn=true negative, fp= false positive and fn=false negative

- F-Measure:

$$F - \text{measure} = \frac{2 * \text{precision} * \text{recall}}{\text{precision} + \text{recall}}$$

Where:

- $\text{Precision} = \frac{\text{tp}}{\text{tp+fp}}$
- $\text{Recall} = \frac{\text{tp}}{\text{tp+fn}}$

In the following figure we report the values of the performances indexes for the different configuration as explained previously.

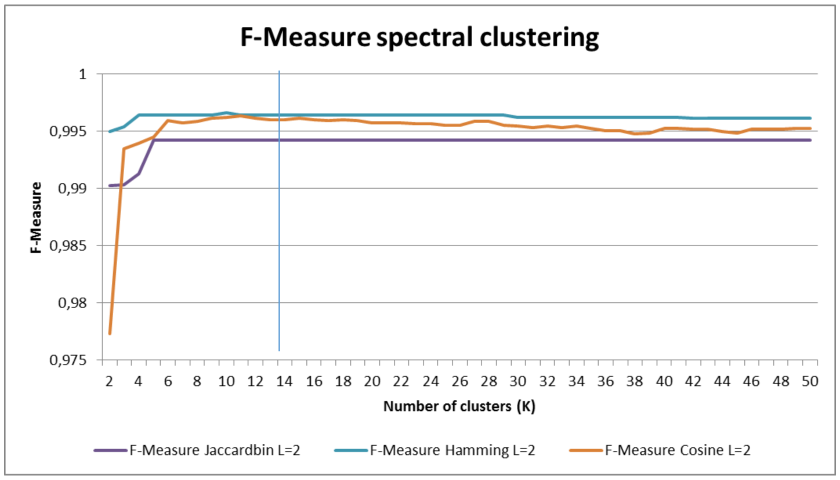

Figure 28 F-measure spectral clustering.

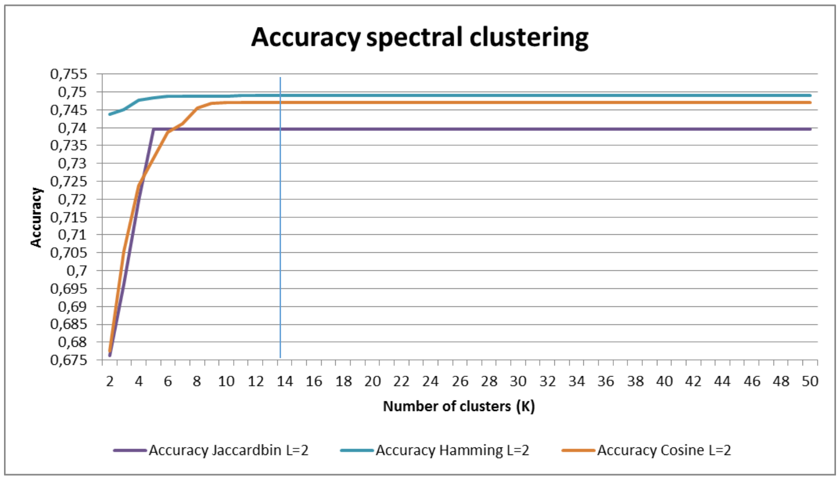

Figure 29 Accuracy spectral clustering.

Analyzing the plots depicted into Figure 28 and Figure 29 and taking into consideration the previous conclusion, we could state that the configuration with the best performances is:

- Distance metric M= Hamming distance
- Number of eigenvectors L = 2
- Number of clusters K=10, we emphasize that in each of the analysis (analysis for selecting the number of eigenvectors, the comparative analysis between k-mean and spectral clustering, and evaluation of the performances) yields same best value for the number of clusters k=10 (as is indicated with vertical lines in Figure 28 and Figure 29).

6.3. Models update

As describe in subsection 5.6.2, the Micro-environments training activity is performed by LTR with a traditional learning base approach (Figure 22). However, using a traditional learning base approach the threat detection modules (i.e. Micro-environments) are trained only once and in this way they reflect only what is present in the training set. Moreover in this approach the users are not involved in the training phase with the disadvantage of not being able to enrich the Micro-environments with the user (possibly expert user) experience.

6.3.1. User-in-the-loop approach

The main idea that is the basis of the “User-in-the-loop” learning base approach is the possibility to customize the Micro-environments on the base of the user expectations. The aim of this approach is to train threat detection modules that try to approximate the user experience in order to supply to the user a threat detection module that reflected their expectations.

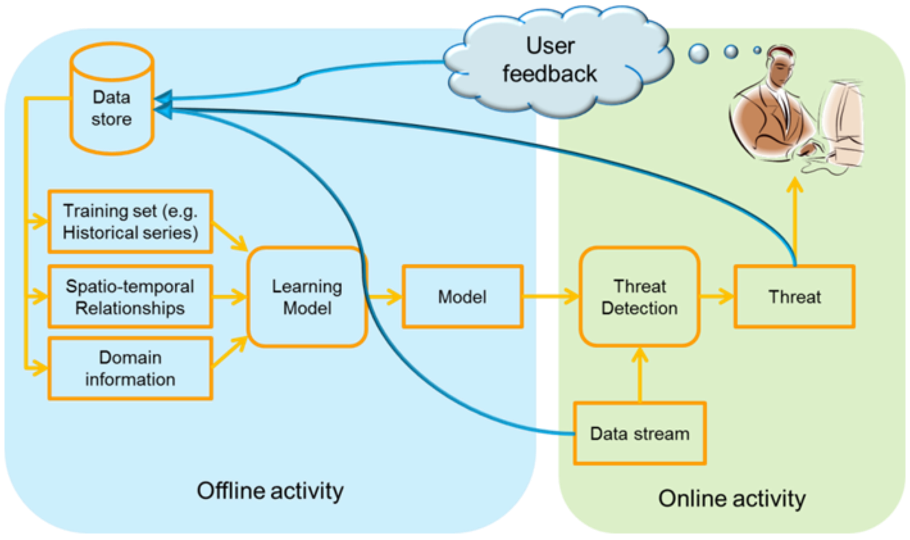

Figure 30 User-in-the-loop learning approach.

As it is possible to see in Figure 30 during the online activity of the system, several information is stored in order to be reused in the following periodic Micro-environment retraining activities. In particular, the new information that is collected is:

- The data stream provided by sensors (i.e. the detected events)
- The threat notifications provided by Micro-environments
- The user feedback (typically operational user feedback)

The collection of this information allows the implementation of an extended learning base approach in which Micro-environments are able to adapt their behavior (i.e. change/modify their threat detection strategy) on the bases of:

- Changing on sensors detection capabilities; If new type of sensors (i.e. new device) or signals analysis algorithms (i.e. video analysis) are introduce into the system, they can produce events that are not covered by the initial training set.
- User expectation; Users can express their opinion specifying if a threat notification is a correct threat notification (i.e. true positive) or it is a wrong threat notification (false positive). Note that a true positive does NOT mean that there has been a terrorist action. Instead, it means that a recognized event is properly turned into a threat notification, according to the expectation of an experienced user.

6.3.2. **Micro-environments Self-Adaptive System**

The extended learning base approach can be considered as a self-adaptive system. Different definitions are present in literature for the self-adaptive system concept, but the most significant for this case is the ones present in [Naqvi, 2012]. A self-adaptive system consists of a closed-loop system that changes its behavior according to changing on user requirements, system properties and environmental characteristics. As we can speak of “Micro-environments self-adaptive system”, in this subsection an analysis of the self-adaptive goals, the modelling dimensions and the engineering approach of the Micro-environments self-adaptive system is reported.

The self-adaptive goals

The main goals of the Micro-environments self-adaptive system can be summarized in:

- Provide to the users some threat detection modules (i.e. Micro-environments) that reflect their expectations
- Update the Micro-environments threat detection strategies according to the collected user feedback
- Increase the Micro-environment’s threat detection performance

If we try to analyze the goals dimensions it is possible to see that:

- In this system, a dynamic evolution is required. The models update activity must be constantly held since, sensor capabilities and user expectations may change over time.

- The system has a constrained flexibility value because the models update activity is strongly connected with the feedback collection activity.
- The duration of these goals is persistent because for the entire life of the system these goals must be guaranteed.
- There are multiple goals that are dependent to each other.

Causes of Self-Adaptivity Dimension

The origins of the changes in the Micro-environments self-adaptive system are two, the change of sensor capabilities and the feedback provided by user during the interaction with the system.

The change of sensor capabilities is an internal cause that is due to hardware replacement or introduction of new analysis algorithms. The type of this change is technological because new hardware or new software is introduced. The frequency of this change can be frequent (in order of months) and it can be foreseeable.

The change due to the user feedback is an external and non-functional case which aims to increase the Micro-environments performance. Also this change can be frequent (in the order of weeks) and foreseeable.

Reaction Mechanisms Dimension

This section wants analyze the system's reaction towards the change. As described before, in this case study we have:

- The change of sensor capabilities that is:
  - A hybrid change because it can influence both of parametric and structural fields of the system
  - Assisted because there is the human influence/interaction
  - Decentralized and local because it can be distributed among several components
  - It has a short duration because the replacement of one or more sensors requires at most few hours
  - The adaptation is event-based.
- The change due to the user feedback that is:
  - Parametric because in this change only some HMM parameters are updated
  - Assisted because during the interaction with the system, users provide their feedback
  - Centralized and local because during the Micro-environments update activity, each Micro-environment is updated independently
  - It has a medium duration because the collection activity of users feedback takes several months
  - The adaptation can be event-based (e.g., a user starts the threat detection models update activity) or time-based (e.g., every two months the system triggers the threat detection models update activity).

Effects Dimension

The effects dimension captures the impact of the adaptation on the system. If we try to analyze the impact on the system in case of some fails on the self-adaptivity activity (i.e., the criticality), we can note that for both the changes the two activities are quite critical because in the first case (hardware change) some Micro-environments can receive some information that they cannot analyzed, while in the second change users cannot customized each Micro-environments on the bases of their expectations. Both of the changes are non-deterministic because the consequences of self-adaptivity cannot be predicted in value and time.

The overhead that occurred during the adaptation on the system’s performances in both of the change is quite insignificant (the models update activity is an off-line activity that not infect the online activity of the system).

6.3.3. **Proposed model update approach**

The proposed approach for the Micro-environments update activity is based on the user feedback. As it is possible to see in Figure 24 and Figure 30 an adaptive feedback control loop is proposed.

During the online activity of the system each Micro-environments analyze the data stream provided by sensors according to their threat detection strategy. When a Micro-environment identifies that an event is a potential symptom of a threat situations, it sends a threat notification at the user. At this point the user analyzes this notification and all this information (detected event, threat notification and user feedback) are stored into the database in order to be used for the models update activity.

The models update activity can be triggered in two ways:

- Automatically from the system
- Manually from the user.

The system can start the update activity according to a time base strategy (e.g. every two months) or according to a performance strategy (e.g. if the last twenty user feedback express that last twenty threat notifications are wrong then start the models update activity) while the user in every time can decide to start the update activity.

7. Medium Term reasoning layer

As mentioned in Section 4.1.3, TENSOR aims to provide help at different actors and users of the system. In this section, the last component (i.e. the Medium Term Reasoning) is presented.

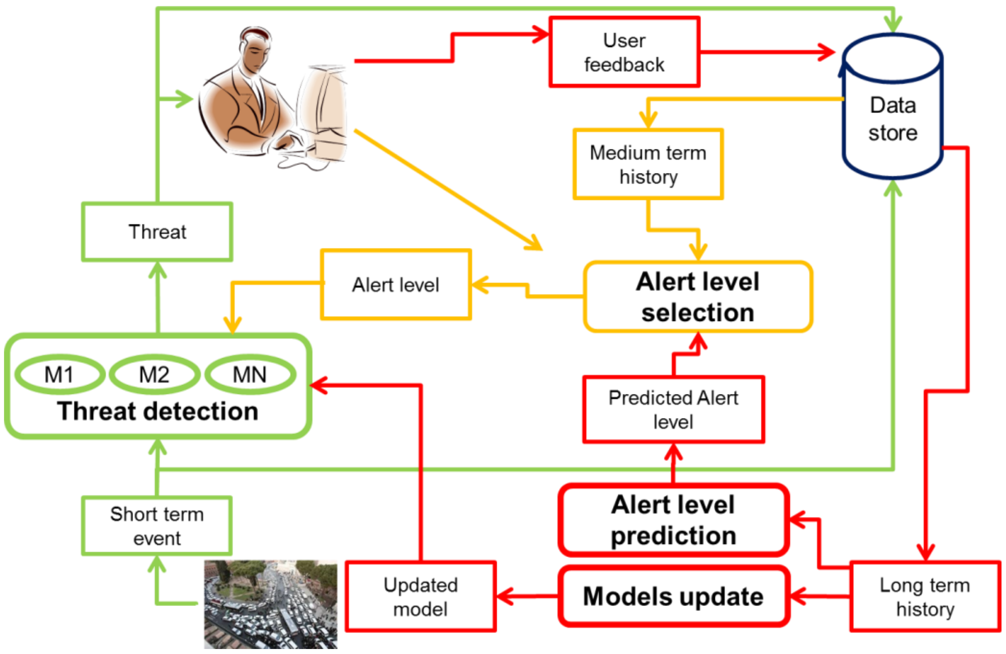

Figure 31 Medium Term Reasoning (orange lines).

The MTR module supports overall decision making by re-evaluating the alert level within a longer temporal scope than STR module and shorter temporal scope than LTR module. It considers the used alert level (STR), the predicted alert level (LTR), the symbolic and threat events and when available, feedback from domain expert users regarding the alert level (graphical interface).

The output of the MTR analysis and the proposed alert levels are shown at tactical users throw a GUI. Note that the output of this module is only a proposal, the tactical user can confirm or change the proposal according to his experience. In this sense, the MTR works like a Decision Support System (DSS) that presents the information necessary at the user to make his choice and it also provides a proposal solution according to the output of its analysis.

7.1. Decision Support System overview

In general, a DSS is a computer system used to give support, more than to automate, the decision making process. The support to a decision means to help the people that work alone or in groups to gather intelligence, to generate alternatives and to take decisions. Supporting the decision making process implies the support of the estimation, the evaluation and/or the comparison between alternatives. In practice, references to DSS are usually references to computer applications that realize a support function.

The decision support systems term has been used in very different ways and has been defined in different ways attending to the point of view of the author. Some of the most cited ones are:

- "DSS is an interactive computer-based system that helps decision makers to utilize data and models to solve unstructured problems" [Sprague and Carlson, 1982].
- "DSS is a computer-based system that aids the process of decision making" [Finlay, 1994].
- "DSS is an interactive, flexible, and adaptable computer-based information system, especially developed for supporting the solution of a non-structured management problem for improved decision making" [Turban, 1995].

Considering above definitions, DSS ranges from systems answering of simple queries to systems modeling of a complex human decision making process. Therefore, it is easy to put a variety of information systems into the DSS class.

A DSS supports users who have to make decisions at any level of management [Power et al., 2015], whether individuals or groups, semi-structured situations and in informal, through the combination of human judgment and objective information:

- Supports multiple interdependent or sequential decisions.
- Offers assistance in all phases of decision-making process-intelligence, design, selection, and implementation, as well as a variety of processes and decision-making styles.
- It is adaptable by the user at the time to deal with changing conditions.
- Generates learning, resulting in new demands and refinement of the application, which in turn results in further learning.
- Generally uses quantitative models (standard or custom made).
- DSS are equipped with an advanced knowledge management component that enables effective and efficient solution of complex problems.
- Can be implemented for use in web or desktop environments on mobile devices.
- Allows easy implementation of sensitivity analysis.

As with the definition, there is no universally accepted taxonomy for DSS. Different authors propose different classifications. Using the method of assistance as a criterion, [Power et al., 2015] distinguishes between:

- **Model-driven** DSS's: Emphasis is placed on the access and manipulation of a statistical model, financial, optimization or simulation. Use data and parameters provided by users to assist decision makers in the analysis of a situation, which are not necessary intensive data.
- **Communication-driven** DSS's: They have support for multiple people working on the same shared task.
- **Data-driven** DSS's: Also called data-oriented, emphasize access and manipulation of time series of internal company data and sometimes also external data.
- **Documents-driven** DSS's: Manage, retrieve and manipulate unstructured information in a variety of electronic formats.
- **Knowledge-driven** DSS's: They provide experience in the form of facts, rules, procedures, or similar structures specialized for solving problems.

While, if we consider the relationship with the user, [Haettenschwiler, 2001] divides DSS into three groups:

- **Passive** DSS – aids the process of decision making, but cannot bring out explicit decision suggestions or solutions.
- **Active** DSS – brings out explicit decision suggestions or solutions.
- **Cooperative** DSS – allows the decision maker to modify the decision suggestions provided by the system. The process is then repeated until a satisfying solution is generated.

As it happen for the definition and taxonomy of decision support systems, also for the architecture different authors identify different components of a DSS. For example in [Power et al., 2015] a DSS has four basic components:

- The user's interface.
- The data base.
- The analytic and modelling tools.
- The DSS's architecture and net.

While in [Haettenschwiler, 2001] five components were identified:

- Users: With different roles or functions in the decision making process (decision maker, consultants, domain experts, system experts, data collectors).
- Decision context: Must be specific and definable.
- Target system: This describes most of the preferences.
- Knowledge basis: Composed of external data sources, knowledge databases, working databases, data warehouses and meta-databases, mathematical models and methods, procedures, inference and search engines, administrative programs, and systems reports.
- Work environment: For the preparation, analysis and documentation of decision alternatives.

7.2. The developed approach

As described before in section 3.2.3 CEP solutions are aimed at fulfilling the need for continuous processing of streaming data in near real-time generating relevant insights and enabling immediate reactions. Since the role of MTR reasoning components in TENSOR is to process numerous incoming events types of different nature and update the sensitivity of STR in near real time manner, we believe that CEP solution perfectly fits requirements for MTR module. Overall from the functional point of view we see the developed framework as a CEP solution with predictive analytics "soft rules" implemented through STR and LTR modules and declarative rules implemented through the MTR module.

Event patterns specify complex relationships among input events entering the system, their role is to define how incoming events should be processed to detect relevant information (e.g. change of alert level, occurrence of a threat). A classical approach is to put the responsibility for discovering, defining and monitoring complex event patterns on the human domain expert. This can be done either manually based on domain knowledge or by using external tools to discover patterns in data and represent in one of the event processing languages. As an example if a domain expert already knows examples of suspicious sequence of detected actions that might represent a threat in the referred scenario and needs to encode them in a format that can be processed by CEP systems, however this is a very difficult and time-consuming approach.

Hence a domain expert might be offered tools that ease exploration of large datasets in order to help him define processing rules. In [Moraru et al., 2012] and [Kenda et al., 2013] the authors focus on the domain expert and easing the rule generation and verification process for environment sensor data. A unifying user interface displays semantically annotated sensor data from multiple sources and numerous aggregates for easing exploration of achieved data, rule generation, testing and export.

The main difference between this scenario and the thesis scenario is the readily availability of data (environmental sensor measurements or credit card transaction data) for rule discovery and validation as opposed to sensor events representing abnormal actions that constitute a complex event of a terrorist threat and hence appropriate historical data for rule validation.

Alternatively in [Widder et al., 2007] unknown event patterns are found by analyzing the event cloud of an organization, discriminative analyses is used to detect a suspicious combination of events in the feature space and the pattern can be saved into the database after expert validation.

We adapt approach of using statistical machine learning for discovering patterns in incoming event stream, since the event types coming into the framework can be both from multiple sensors (including a “human” sensor) and numerous types (detected actions) and carefully crafting rules in a terrorist scenario is a burdensome process even for an expert. Additional statistical modelling of sets of suspicious actions into a threat event enables “soft” rules that adapt with time and are able to follow the shift in behavior of terrorist groups (the STR module).

Hence multiple types of user action event streams from different sensors are entering STR module which in turn is producing complex events of threat detection. Since the decisions are made using automatically discovered soft rules, there is an intrinsic level of data uncertainty accompanying the generated primary complex events for detecting threat situations. The detected primary complex event is both sent to event consumers and put back into the framework for recursive processing. This type of event can be denoted as internal event according to [Adi and Etzion, 2004], who differentiates between external events in pushed into the system by external sources in runtime and internal events inferred by the system when a particular situation occurs.

In order to enable the proactive prevention of the system rather than just reaction we combine the STR for threat detection with LTR through our MTR module. The LTR module aims at reasoning about the event by using external knowledge sources relevant to the domain (as described in the previous section) and exploiting accumulated historical information from the predefined time period to position the alertness / sensitivity of the overall reasoning module STR to incoming unusual events.

A general functional view of a CEP system is illustrated in Figure 32. The receiver is responsible for receiving events from multiple event streams and passing them along and for modeling solutions for periodic input processing (connection to the clock).The decider, processes events passed on by the receiver and on arrival of a new event checks the set of predefined rules. When a corresponding event pattern is detected appropriate action from the triggered rule is passed to the producer who generates an action (e.g. alarm, security threat). The generated complex events are either sent to event sinks (user components) or return back into the system for further processing. Optional static knowledge about source events can be stored in the knowledge base (e.g. if a “earthquake threat alarm” event has been triggered for an area geographically close to a sea front then additional “tsunami alert” event can be issued).

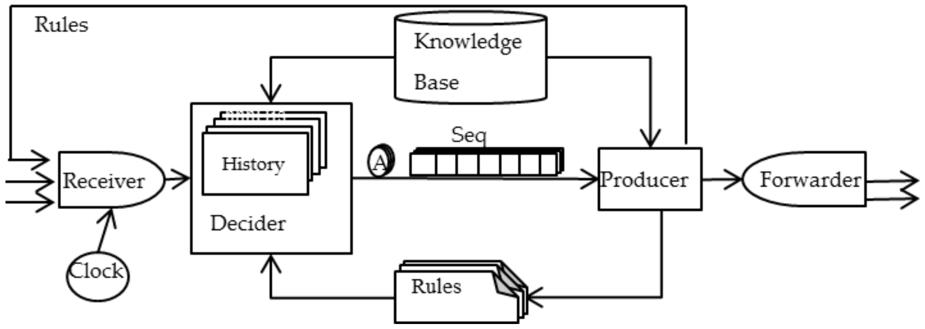

Figure 32 Functional CEP system architecture.

A combination of CEP and predictive analytics is considered in [Fülöp et al., 2012], where classical CEP system [Esper Version 4.11.0] has been extended with decision trees approaches. The authors use complex events produced by rules of the CEP system to label streams of data, generate training sets and statistically model prediction of new complex events.

In this thesis the inputs into the STR module are symbolic representations of unusual activities, sensor-independent events about abnormal situations (e.g., Person Observing Building, People Not Fitting In Environment). These events are produced by the low-level fusion algorithms (see section 4.1) which translate detected sensor inputs into symbolic sensor-independent representations. Furthermore the events are processes with a “soft filter” of the STR component and a complex event, threat event / or not is generated (see Table 12). The LTR reasoning also acts as a “soft filter” (Table 7 Table 13) for aggregated threat data corresponding to a certain “long” period (~1 day) coming from the STR/database and produces predictions for criticality level based on rules (models) trained over long-term historical data and/or external data sources. The MTR module function as the core part of a CEP system, based on a set of predefined rules it makes a decision about the update of the alert level event that will be passed to the STR module. This decision is based upon combining the predicted alert level events coming from the LTR and the used alert level by STR modules (which in turn can be either set by a domain expert user through GUI or a result of the previous MTR phase) as well as the threat events coming from STR.

Table 12 Types of rules from a functional perspective as a CEP engine.

Framework modules	Input source origin → Data type	Rule type and origin → Update periods	Output type → Destination
STR	Sensors → Streams of symbolic events MTR GUI → User alert level MTR → Updated alert level	Short-term "soft" rules (Micro – environments) → Model update (performed by LTR) once a months	Detected threat events -> MTR GUI Used alert level -> MTR GUI
MTR	Data store → Symbolic events and threat probabilities STR → current alert level LTR→ Predicted alert level	Medium-term declarative rules → Generated by domain expert /updated manually or by use of learning algorithms when user feedback available	Updated alert level → STR & MTR GUI
LTR	Data store→ Symbolic events and threat probabilities Expert user→ User feedback	Short-term "soft" rules → Model update (performed by LTR) once a months	Predicted alert level→ MTR & MTR GUI

For the implementation of the MTR module we used Drools an open source project written in Java, licensed under the Apache License, Version 2.0 [Drools 6.01]. The Drools rules engine started as Production Rule System (PRS) rule engine based on the Rete algorithm [Forgy, 1982]. The production rule approach has two parts <when: conditions - then: actions> : the inference engine which performs pattern matching of facts (stored in the working memory) against rules (stored in the production memory) and infers conclusions which consequently result in actions. This is in compliance with the requirements for our MTR CEP component.

Current Drools rule engine uses two algorithms PHREAK and ReteOO originating from Rete. ReteOO is an optimized implementation for object oriented systems and includes several schemes for reasoning with imperfect information, and configuration parameters for the architecture [Sottara et al., 2010]. PHREAK a goal oriented algorithm where partial matching is aggressively delayed, it inherits the enhancements from ReteOO while already being faster and more scalable [Drools 6.01]. Drools is defined as a Hybrid Reasoning System that combine forward chaining (reactive and data driven; a fact is being propagated through the rules until we reach a conclusion) and backward chaining (goal-driven derivation queries e.g. Prolog engine; an engine starts from a conclusion which it tries to satisfy, if this is not feasible it searches for conclusions that it can satisfy “subgoals”). Two main modules are used: the rule engine (Drools Expert) and the module enabling the event processing capabilities (Drools Fusion).

Drools Expert is a framework to build expert systems, it is a declarative, rule based coding environment, which is used to define, execute and maintain the rules. Additionally to declared rules a complete expert systems must also include domain knowledge in the form of an ontological model.

Drools Fusion module adds event processing capabilities as identifying temporal relationships between events, composite and aggregated events, furthermore the rule engine can be now perceive as operating on events and not only on facts. Traditional Rete algorithm has been enhanced with to provide features such as aggregation of values in time-based windows [Walzer et al., 2008]. This makes Drools Fusion suitable to implement standard Complex Event Processing scenarios.

Table 13 shows the set of rules and parameters used to generate a complex event of updating the alert level by the MTR module in function of the input event streams. The module currently simulates virtual sensors that produce all incoming events.

Table 13 MTR update Alert Level (AL).

AL LTR / **AL STR**	**Low**	**Medium**	**High**
Low	**when**: <over window : time (N) threat events are < T1 % of all events> **then** : <AL:= Low>	**when**: <over window : time (N) threat events are < T2 % of all events> **then**: <AL Low>	**when**: <over window : time (N) threat events are < T3 % of all events> **then** : <AL:= Low>
	when: <over window : time (N) threat events are >= T1 % of all events> **then** : <AL:= Medium>	**when**: <over window : time (N) threat events are >= T2 % of all events> **then**: <AL:= Medium>	**when**: <over window : time (N) threat events are >= T3 % of all events> **then** : <AL:= Medium>
Medium	**when**: <over window : time (N) threat events are < (1-T2) % of all events> **then** : <AL:= Low>	**when**: <over window : time (N) threat events are < (1-T1) % of all events> **then** : <AL:= Low >	**when**: <over window : time (N) threat events are < =MIN % of all events> **then** : <AL:= Low >
	when: <over window : time (N) threat events >= (1-T2)% & < MAX% of all events > **then** : <AL:= Medium>	**when**: <over window : time (N) threat events >= (1-T1) % & < T1% of all events > **then** : <AL:= Medium >	**when**: <over window : time (N) threat events > MIN% & < T2% of all events > **then** : <AL:= Medium >
	when: <over window : time (N) threat events are >= MAX% of all events> **then** : <AL:= High>	**when**: <over window : time (N) threat events are >= T1 % of all events> **then** : <AL:= High>	**when**: <over window : time (N) threat events are >= T2 % of all events> **then** : <AL:= High>
High	**when**: <over window : time (N) threat events are < (1-T3) % of all events> **then** : <AL:= Medium >	**when**: <over window : time (N) threat events are < (1-T2) % of all events> **then** : <AL:= Medium >	**when**: <over window : time (N) threat events are < (1-T1) % of all events> **then** : <AL:= Medium >
	when: <over window : time (N) threat events are >= (1-T3) % of all events> **then** : <AL:= High>	**when**: <over window : time (N) threat events are >= (1-T2) % of all events> **then** : <AL:= High>	**when**: <over window : time (N) threat events are >=(1-T1) % of all events> **then** : <AL:= High>

The parameters at hand are MIN, T1, T2 , T3, MAX and N and the variables of interest are the alert level $\mathrm{AL} = \{\mathrm{Low}, \mathrm{Medium}, \mathrm{High}\}$ for STR, LTR, and MTR modules respectively $\mathrm{AL}_{\mathrm{STR}}$, $\mathrm{AL}_{\mathrm{LTR}}$, $\mathrm{AL}_{\mathrm{MTR}}$. The parameters can be explained in the following manner:

- T1 is the parameter of the condition, that defines the minimum needed presence of threat events among all events for a window of length N in case when $\mathrm{AL}_{\mathrm{STR}}$ and $\mathrm{AL}_{\mathrm{LTR}}$ have same values, so that the $\mathrm{AL}_{\mathrm{MTR}}$ coming out of MTR and being sent to STR is a one-step incrimination of $\mathrm{AL}_{\mathrm{STR}}$.
- T2 is the parameter of the condition, that defines the minimum needed presence of threat events among all events for a window of length N in case when $\mathrm{AL}_{\mathrm{STR}}$ has a value one-step less than $\mathrm{AL}_{\mathrm{LTR}}$, so that the $\mathrm{AL}_{\mathrm{MTR}}$ coming out of MTR and being sent to STR is a one-step incrimination of $\mathrm{AL}_{\mathrm{STR}}$.
- T3 is the parameter of the condition, that defines the minimum needed presence of threat events among all events for a window of length N in case when $\mathrm{AL}_{\mathrm{STR}}$ has a value two steps less than $\mathrm{AL}_{\mathrm{LTR}}$, so that the $\mathrm{AL}_{\mathrm{MTR}}$ coming out of MTR and being sent to STR is a one-step incrimination of $\mathrm{AL}_{\mathrm{STR}}$.

All the other relations and conclusions from Table 13 are extrapolated of the above three statements, additionally MAX > T1 > T2 > T3 > MIN and in our initial setting MIN=0, T1=90, T2=80, T3=60, MAX=100.

In case $\mathrm{AL}_{\mathrm{STR}}$ was set by a user through GUI interface within a time window N, the rules defined in Table 13 are ignored and the current alert level is maintained until appropriate conditions are met.

However, in the presented set of rules, there is no consideration about spatial constraints present between different physical monitored environments. Examples of these constrains referred to the validation scenario (Figure 23) can be "What happens in the Ministry of Finance influences also the Square" or "What happens at the Ministry of Finance does not influence the Church". These constraints, for example, can be caused by:

- Proximity between environments (in the validation scenario the Square and the Ministry are close each other while the Church is far from them)
- Affiliation to the same type of environment (e.g., if in a metro station occurred a condition that triggers the rule that set the "High" alert level, in all the other monitored metro stations the alert level must be increase of one level).

During the alert level selection activity, the MTR starts with the evaluation rules for the time series analysis (Table 13) and only when this phase is finished for all the STR Micro-environments the MTR starts to evaluate the spatial constraints rules (Table 14).

Table 14 Example of spatial constraints rules.

Square AL	Ministry Low AL	Ministry Medium AL	Ministry High AL
Low	**when**:<Ministry_{AL} = Low > **then** : <AL:= Low >	**when**:<Ministry_{AL} = Medium > **then** : <AL:= Medium >	**when**:<Ministry_{AL} = High > **then** : <AL:= Medium >
Medium	**when**:<Ministry_{AL} = Low > **then** : <AL:= Medium >	**when**:<Ministry_{AL} = Medium > **then** : <AL:= Medium >	**when**:<Ministry_{AL} = High > **then** : <AL:= High >
High	**when**:<Ministry_{AL} = Low > **then** : <AL:= High >	**when**:<Ministry_{AL} = Medium > **then** : <AL:= High >	**when**:<Ministry_{AL} = High > **then** : <AL:= High >

In Figure 33 it is possible to see, the information that the MTR shoes at tactical users. In particular for each monitored environments the MTR shoes:

- The current alert level
- The alert level proposed by LTR
- The percentage of detected threats occurred in the last medium term time window
- and, in the end, a proposed alert level inferred from the rules analysis.

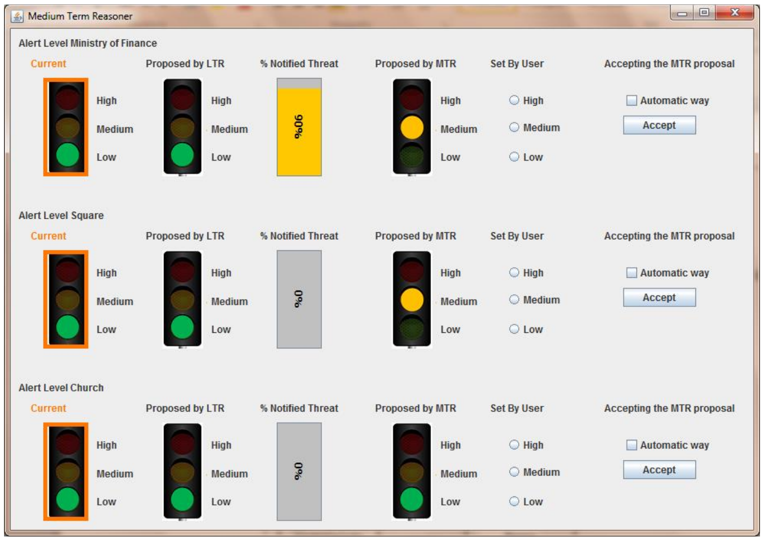

Figure 33 Medium Term Reasoner GUI.

As describe before, the MTR does not replace the expert users, in the right part of Figure 33 there are graphical component that give at the user the possibility to:

- Accept the MTR proposed alert levels (i.e. update or confirm the current alert level used by Micro-environment)
- Manually set the alert levels of each Micro-environments according to his experience
- Accept in "an automatic way" the MTR updates made to the alert levels.

7.3. Improvements for the MTR module

As mentioned another major bottleneck of CEP systems is the need for manual definition of rules and parameters defining relationships between primitive and/or complex events. Incorrect definitions of these handcrafted rules and parameters can lead to incorrect detection of complex events and false or missed warnings. As an example the authors in [Turchin et al., 2009] propose a generic framework for automating both the initial definition and the update of the rules over time based on an online learning with Discrete Kalman Filters. The method was tested in a computer network intrusion detection scenario but needs a constant feedback from the expert to verify the performance and to update the rules. Since the role of the domain expert users is crucial for the functioning of the developed framework, a similar algorithm can be used for the MTR module. Future work will also investigate ways to more efficiently tie together the predictive analytics part and the CEP system in our implementation.

8. Conclusion

This thesis has shown it's possible to fuse historical information on the behavioral patterns of terrorist groups in different socio-economical conditions with short term information like that of a monitoring system resulting in a better threat detection. To obtain this result different machine learning methods have been shown to be effective at different time frames and for heterogeneous data sources.

Actual data were video feeds but TENSOR's design and software architecture allows to incorporate also other digital traces of terrorist activities like social networks, financial transactions or individual mobility patterns.

TENSOR's machine learning methodological kernel could be implemented with minor redesign and significant performance improvement using some of the frameworks discussed in 3.3.2 like a Lamba architecture and tools like Flink or H2O.

At the time the system was designed most of these tools were not available moreover the software architecture of TENSOR was constrained by the customer requirements from LEA and design choices of the PROACTIVE European project.

From the viewpoint of LEA requirements, TENSOR fits into the vision emerged in the very last month that to fight effectively terrorist activities, at least in Europe, a system must look at not only the short term activity but at the wider picture of how groups behave and react to different socio-political conditions and gear up to stage concentrated large scale attacks.

Acknowledgements

Part of this work has been carried out in the scope of the PROACTIVE project (FP7-285320) (http://www.proactive-project.eu). The author acknowledge help and contributions from all partners of the project.

References

- Adi, A., & Etzion, O. (2004). Amit-the situation manager. The VLDB Journal—The International Journal on Very Large Data Bases, 13(2), 177-203.
- Allanach, J., Tu, H., Singh, S., Willett, P., & Pattipati, K. (2004, March). Detecting, tracking, and counteracting terrorist networks via hidden Markov models. In Aerospace Conference, 2004. Proceedings. 2004 IEEE (Vol. 5). IEEE.
- Andersson, M., & Johansson, R. (2010, November). Multiple sensor fusion for effective abnormal behaviour detection in counter-piracy operations. In Waterside Security Conference (WSS), 2010 International (pp. 1-7). IEEE.
- Anicic, D., Fodor, P., Rudolph, S., Stühmer, R., Stojanovic, N., & Studer, R. (2011). Etalis: Rule-based reasoning in event processing. In Reasoning in event-based distributed systems (pp. 99-124). Springer Berlin Heidelberg.
- Arasu, A., Babu, S., & Widom, J. (2006). The CQL continuous query language: semantic foundations and query execution. The VLDB Journal—The International Journal on Very Large Data Bases, 15(2), 121-142.
- Atefeh, F., & Khreich, W. (2015). A survey of techniques for event detection in twitter. Computational Intelligence, 31(1), 132-164.
- Azimirad, E., & Haddadnia, J. (2015). The Comprehensive Review On JDL Model In Data Fusion Networks: Techniques and Methods. International Journal of Computer Science and Information Security, 13(1), 53.
- Baxter, K., Courage, C., & Caine, K. (2015). Understanding Your Users: A Practical Guide to User Research Methods. Morgan Kaufmann.
- Bekkerman, R., Bilenko, M., & Langford, J. (Eds.). (2011). Scaling up machine learning: Parallel and distributed approaches. Cambridge University Press.
- Bennett, B. T. (2007). Understanding, assessing, and responding to terrorism: protecting critical infrastructure and personnel. John Wiley & Sons.
- Brandt, P. T., & Williams, J. T. (2007). Multiple time series models (No. 148). Sage.
- Bruzzone, A., Tremori, A., & Massei, M. (2009). Serious games for training and education on defense against terrorism. GENOA UNIV (ITALY).
- Caspian Learning (2008). Serious games in defence education. Sunderland, Royaume-Uni.
- Chakravarthy, S., & Mishra, D. (1994). Snoop: An expressive event specification language for active databases. Data & Knowledge Engineering, 14(1), 1-26.

- Cheong, M., & Lee, V. C. (2011). A microblogging-based approach to terrorism informatics: Exploration and chronicling civilian sentiment and response to terrorism events via Twitter. Information Systems Frontiers, 13(1), 45-59.
- Chung, W. (2012, June). Categorizing temporal events: A case study of domestic terrorism. In Intelligence and Security Informatics (ISI), 2012 IEEE International Conference on (pp. 159-161). IEEE.
- Clauset, A., & Woodard, R. (2013). Estimating the historical and future probabilities of large terrorist events. The Annals of Applied Statistics, 7(4), 1838-1865.
- Coffman, T. R., & Marcus, S. E. (2004, March). Dynamic classification of groups through social network analysis and hmms. In Aerospace Conference, 2004. Proceedings. 2004 IEEE (Vol. 5, pp. 3197-3205). IEEE.
- Crenshaw, M. (1981). The causes of terrorism. Comparative politics, 13(4), 379-399.
- Cugola, G., & Margara, A. (2012). Processing flows of information: From data stream to complex event processing. ACM Computing Surveys (CSUR), 44(3), 15.
- Curtis, P., Harb, M., Abielmona, R., & Petriu, E. (2016). Classification-Driven Video Analytics for Critical Infrastructure Protection. In Recent Advances in Computational Intelligence in Defense and Security (pp. 45-69). Springer International Publishing.
- D'Orazio, V., Yonamine, J. E., & Schrodt, P. A. (2011, March). Predicting intra-state conflict onset: An event data approach using euclidean and levenshtein distance measures. In 69th Annual Meeting of the Midwest Political Science Association, Chicago, Il, USA.
- Dawoud, K., Jarada, T. N., Almansoori, W., Chen, A., Gao, S., Alhajj, R., & Rokne, J. (2013). Data Analysis Based Construction and Evolution of Terrorist and Criminal Networks. In Handbook of Computational Approaches to Counterterrorism (pp. 301-321). Springer New York.
- De Simio, F., Tesei, M., & Setola, R. (2016). Game Theoretical Approach for Dynamic Active Patrolling in a Counter-Piracy Framework. In Recent Advances in Computational Intelligence in Defense and Security (pp. 423-444). Springer International Publishing.
- Department of Defense, Joint Publication 1-02 Dictionary of Military and Associated Terms, 2010. [Online]. Available: htttp://www.dtic.mil/doctrine/new_pubs/jp1_02.pdf
- Desmarais, B. A., & Cranmer, S. J. (2013). Forecasting the locational dynamics of transnational terrorism: A network analytic approach. Security Informatics, 2(1), 1-12.
- Dixon, S. J., Dixon, M. B., Elliott, J., Guest, E., & Mullier, D. J. (2011b). A Neural Network for Counter-Terrorism. In Research and Development in Intelligent Systems XXVIII (pp. 229-234). Springer London.
- Dixon, S., Guest, E., Dixon, M. B., Elliot, J., & Mullier, D. (2011a). DScent Final Report.
- Dymarski, P. (2011). Hidden Markov Models, Theory and Applications. InTech Open Access Publishers.
- Eiselt, H. A., Bhadury, J., & Burkey, M. L. (2013). An optimization-based framework for modelling counter-terrorism strategies. OR Insight, 26(1), 7-31.
- Enders, W., & Sandler, T. (2002). Patterns of transnational terrorism, 1970–1999: alternative time-series estimates. International Studies Quarterly, 46(2), 145-165.
- Enders, W., & Sandler, T. (2011). The political economy of terrorism. Cambridge University Press.

- Enders, W., Sandler, T., & Gaibulloev, K. (2011). Domestic versus transnational terrorism: Data, decomposition, and dynamics. Journal of Peace Research, 48(3), 319-337.
- Endy, C. lim, K. I. Eng and A. S. Nugroho, "Implementation of Intelligent Searching Using Self-Organizing Map for Webmining Used in Document Containing Information in Relation to Cyber Terrorism," Advances in Computing, Control and Telecommunication Technologies (ACT), 2010 Second International Conference on, Jakarta, 2010, pp. 195-197.
- Esper Version 4.11.0, EsperTech Inc., Available at: http://www.espertech.com/download/
- Etzion, O., & Niblett, P. (2010). Event processing in action. Manning Publications Co.
- Fan, Wei, and Albert Bifet. "Mining big data: current status, and forecast to the future." ACM sIGKDD Explorations Newsletter 14.2 (2013): 1-5.
- Finlay, P. N. Introducing decision support systems. Oxford, UK Cambridge, Mass., NCC Blackwell; Blackwell Publishers, 1994.
- Foresti, G. L., Farinosi, M., & Vernier, M. (2015). Situational awareness in smart environments: socio-mobile and sensor data fusion for emergency response to disasters. Journal of Ambient Intelligence and Humanized Computing, 6(2), 239-257.
- Forgy, C. L. (1982). Rete: A fast algorithm for the many pattern/many object pattern match problem. Artificial intelligence, 19(1), 17-37.
- Fülöp, L. J., Beszédes, Á., Tóth, G., Demeter, H., Vidács, L., & Farkas, L. (2012, September). Predictive complex event processing: a conceptual framework for combining complex event processing and predictive analytics. In Proceedings of the Fifth Balkan Conference in Informatics (pp. 26-31). ACM.
- Ghahramani, Z. (2004). Unsupervised learning. In Advanced Lectures on Machine Learning (pp. 72-112). Springer Berlin Heidelberg.
- Golestan, K., Soua, R., Karray, F., & Kamel, M. S. (2016). Situation awareness within the context of connected cars: A comprehensive review and recent trends. Information Fusion, 29, 68-83.
- Granstrom, K., Willett, P., & Bar-Shalom, Y. (2015, April). A Bernoulli filter approach to detection and estimation of hidden Markov models using cluttered observation sequences. In Acoustics, Speech and Signal Processing (ICASSP), 2015 IEEE International Conference on (pp. 3911-3915). IEEE.
- GTD (Global Terrorism Database) Codebook: Inclusion Criteria and Variables, December 2015.
- Gupchup, J., Terzis, A., Burns, R., & Szalay, A. (2009). Model-based event detection in wireless sensor networks. arXiv preprint arXiv:0901.3923.
- Ha, T., Lee, S., & Kim, N. (2015). Development of a User-Oriented IoT Middleware Architecture Based on Users' Context Data. In Distributed, Ambient, and Pervasive Interactions (pp. 287-295). Springer International Publishing.
- Haettenschwiler, P. (2001). Neues anwenderfreundliches konzept der entscheidungsunterstützung. Zurich, vdf Hochschulverlag AG, 189-208.
- Hall, D., & Llinas, J. (Eds.). (2001). Multisensor data fusion. CRC press
- Hastie, T., Tibshirani, R., & Friedman, J. (2009). Unsupervised learning (pp. 485-585). Springer New York.

- Hosmer, D. W., Lemeshow, S., & May, S. (2008). Model development. Applied Survival Analysis: Regression Modeling of Time-to-Event Data, Second Edition, 132-168.
- Hummel, G., Russ, M., Stütz, P., Soldatos, J., Rossi, L., Knape, T., ... & Kompatsiaris, I. (2012). Intelligent Multi Sensor Fusion System for Advanced Situation Awareness in Urban Environments. In Future Security (pp. 93-104). Springer Berlin Heidelberg.
- Inkpen, D. (2016). Text Mining in Social Media for Security Threats. In Recent Advances in Computational Intelligence in Defense and Security (pp. 491-517). Springer International Publishing.
- Inyaem, U., Meesad, P., Haruechaiyasak, C., & Tran, D. (2009, December). Ontology-based terrorism event extraction. In Information Science and Engineering (ICISE), 2009 1st International Conference on (pp. 912-915). IEEE.
- Jayanthi, S. K., & Sasikala, M. S. (2011). XGraphticsCLUS: Web mining hyperlinks and content of terrorism websites for homeland security. International Journal of Advanced Networking and Applications, 2(6).
- Jones, S. G., & Libicki, M. C. (2008). How terrorist groups end: Lessons for countering al Qa'ida. Rand Corporation.
- Kenda, K., Fortuna, C., Moraru, A., Mladenić, D., Fortuna, B., & Grobelnik, M. (2013). Mashups for the web of things. In Semantic Mashups (pp. 145-169). Springer Berlin Heidelberg.
- King, G., & Zeng, L. (2001). Logistic regression in rare events data. Political analysis, 9(2), 137-163.
- Klein, L. A. (2004). Sensor and data fusion: a tool for information assessment and decision making (Vol. 324). Bellingham^ eWA WA: Spie Press.
- LaFree, G., & Dugan, L. (2007). Introducing the global terrorism database. Terrorism and Political Violence, 19(2), 181-204.
- Landes, W. M. (1978). An economic study of US aircraft hijacking, 1961-1976. The Journal of Law & Economics, 21(1), 1-31.
- Landset, S., Khoshgoftaar, T.M., Richter, A.N., Hasanin, T., "A survey of open source tools for machine learning with big data in the Hadoop ecosystem", Journal of Big Data (2015) 2:24.
- Mahmood, T., & Rohail, K. (2012, October). Analyzing terrorist incidents to support counter-terrorism-Events and methods. In Robotics and Artificial Intelligence (ICRAI), 2012 International Conference on (pp. 149-156). IEEE.
- Malathi, A., Baboo, D. S. S., Dhanabhakyam, M., Punithavalli, M., Parimala, R., Nallaswamy, R., ... & Sasikala, M. S. (2011). Algorithmic Crime Prediction Model Based on the Analysis of Crime Clusters. Global Journal of Computer Science and Technology, 11(11), 47-51.
- Markman, A., Rachkovskij, D., Misuno, I., & Revunova, E. (2003). Analogical reasoning techniques in intelligent counterterrorism systems.
- Martinez, V., Simari, G., Sliva, A., & Subrahmanian, V. S. (2008). CONVEX: similarity-based algorithms for forecasting group behavior. Intelligent Systems, IEEE, 23(4), 51-57.
- McCarthy, D., & Dayal, U. (1989, June). The architecture of an active database management system. In ACM Sigmod Record (Vol. 18, No. 2, pp. 215-224). ACM.

- McMorrow, D. (2009). Rare events (No. JSR-09-108). MITRE CORP MCLEAN VA JASON PROGRAM OFFICE.
- Mickolus, E. F., Sandler, T., Murdock, J. M., & Flemming, P. (2012). International Terrorism: Attributes of Terrorism Events (ITERATE), 1968–2011. Dunn Loring, VA: Vinyard Software.
- Monekosso D, Remagnino P (2007) Monitoring behaviour with an array of sensors. Comput Intell 23(4):420–438.
- Moraru, A., Kenda, K., Fortuna, B., Bradeško, L., Škrjanc, M., Mladenić, D., & Fortuna, C. (2012). Supporting rule generation and validation on environmental data in EnStreaM. In The Semantic Web: ESWC 2012 Satellite Events (pp. 441-446). Springer Berlin Heidelberg.
- Naqvi, M. (2012). Claims and supporting evidence for self-adaptive systems–A literature review.
- Nizamani, S., Memon, N., Wiil, U. K., & Karampelas, P. (2013). Modeling suspicious email detection using enhanced feature selection. arXiv preprint arXiv:1312.1971.
- Norris, J. R. (1998). Markov chains (No. 2008). Cambridge university press.
- Petris, S., Georgoulis, C., Soldatos, J., Giordani, I., Sormani, R., & Djordjevic, D. (2014). Predicting Terroristic Attacks in Urban Environments: An Internet-of-Things Approach. International Journal of Security and Its Applications, 8(4), 195-218.
- Pevehouse, J.C., Brozek J., (2010) Time series analysis in political science. In: Box-Steffensmeier J, Brady H, Collier D (eds) Oxford handbook of political Methodology. Oxford University Press, New York.
- Pires, I. M., Garcia, N. M., Pombo, N., & Flórez-Revuelta, F. (2016). From Data Acquisition to Data Fusion: A Comprehensive Review and a Roadmap for the Identification of Activities of Daily Living Using Mobile Devices. Sensors, 16(2), 184.
- Popp, R., Pattipati, K., Willett, P., Serfaty, D., Stacy, W., Carley, K., ... & Singh, S. (2005, March). Collaborative Tools for Counter-Terrorism Analysis. In Aerospace Conference, 2005 IEEE (pp. 1-10). IEEE.
- Porter, M. D., & White, G. (2012). Self-exciting hurdle models for terrorist activity. The Annals of Applied Statistics, 6(1), 106-124.
- Power, D. J., Sharda, R., & Burstein, F. (2015). Decision support systems. John Wiley & Sons, Ltd.
- Raghavan, V., Galstyan, A., & Tartakovsky, A. G. (2013). Hidden Markov models for the activity profile of terrorist groups. The Annals of Applied Statistics, 7(4), 2402-2430.
- RAND (2012) RAND Database of Worldwide Terrorism Incidents (http://www.rand.org/nsrd/projects/terrorism-incidents.html).
- Rao, D. V. (2016). Design and Development of Intelligent Military Training Systems and Wargames. In Recent Advances in Computational Intelligence in Defense and Security (pp. 555-603). Springer International Publishing.
- Reid, E., Qin, J., Chung, W., Xu, J., Zhou, Y., Schumaker, R., ... & Chen, H. (2004). Terrorism knowledge discovery project: A knowledge discovery approach to addressing the threats of terrorism. In Intelligence and Security Informatics (pp. 125-145). Springer Berlin Heidelberg.

- Sabzevar, I. (2015). A COMPREHENSIVE REVIEW OF THE MULTI-SENSOR DATA FUSION ARCHITECTURES. Journal of Theoretical and Applied Information Technology, 71(1).
- Sahito, F., Latif, A., & Slany, W. (2011, September). Weaving twitter stream into linked data a proof of concept framework. In Emerging Technologies (ICET), 2011 7th International Conference on (pp. 1-6). IEEE.
- Sandler, T., & Siqueira, K. (2009). Games and terrorism recent developments. Simulation & Gaming, 40(2), 164-192.
- Sandler, T., Arce, D. G., & Enders, W. (2009). Transnational terrorism. In: Lomberg Bjørn (ed.) Global Crises, Global Solutions, 2nd edn. Cambridge: Cambridge University Press, 516–562.
- Schrodt, P. A. (2000). Pattern recognition of international crises using hidden Markov models. Political complexity: Nonlinear models of politics, 296-328.
- Schrodt, P. A., Yonamine, J., & Bagozzi, B. E. (2013). Data-based computational approaches to forecasting political violence. In Handbook of Computational Approaches to Counterterrorism (pp. 129-162). Springer New York.
- Shahir, H. Y., Glasser, U., Shahir, A. Y., & Wehn, H. (2015, October). Maritime situation analysis framework: Vessel interaction classification and anomaly detection. In Big Data (Big Data), 2015 IEEE International Conference on (pp. 1279-1289). IEEE.
- Shen, Q., & Boongoen, T. (2012). Fuzzy orders-of-magnitude-based link analysis for qualitative alias detection. Knowledge and Data Engineering, IEEE Transactions on, 24(4), 649-664.
- Silverman, B. G., Johns, M., Cornwell, J., & O'Brien, K. (2006). Human behavior models for agents in simulators and games: part I: enabling science with PMFserv. Presence: Teleoperators and Virtual Environments, 15(2), 139-162.
- Singh, S., Allanach, J., Tu, H., Pattipati, K., & Willett, P. (2004, October). Stochastic modeling of a terrorist event via the ASAM system. In Systems, Man and Cybernetics, 2004 IEEE International Conference on (Vol. 6, pp. 5673-5678). IEEE.
- Singh, S., Tu, H., Donat, W., Pattipati, K., & Willett, P. (2009). Anomaly detection via feature-aided tracking and hidden Markov models. Systems, Man and Cybernetics, Part A: Systems and Humans, IEEE Transactions on, 39(1), 144-159.
- Smith, R. (2009). The long history of gaming in military training. Simulation & Gaming.
- Sormani, R., Soldatos, J., Vassilaras, S., Tisato, F., Giordani, I., "A Serious Game empowering the Prediction of Potential Terroristic Actions", Journal of Policing, Intelligence and Counter Terrorism Vol. 11, Iss. 1, 2016.
- Sottara, D., Mello, P., & Proctor, M. (2010). A configurable rete-oo engine for reasoning with different types of imperfect information. Knowledge and Data Engineering, IEEE Transactions on, 22(11), 1535-1548.
- Sprague, R.H. & Carlson E.D., Building Effective Decision Support Systems Englewood Clifts, N.J.: Prentice-Hall, Inc.: 1982, ISBN: 0-13-086215-0.
- Susi, T., Johannesson, M., & Backlund, P.. Serious games: An overview. Technical report HIS-IKI-TR-07-001 University of Skovde (2007).

- The JBoss Drools team. Drools Expert User Guide Version 6.0.1. Final (20.12.2013). http://docs.jboss.org/drools/release/6.0.1.Final/drools-docs/html/index.html
- Tinguriya, D., & Kumar, B. (2010). Detecting terror-related activities on the web using neural network. Oriental Journal of Computer Science and Technology, 3(2), 6.
- Turban, E. Decision support and expert systems: management support systems. Englewood Cliffs, N.J., Prentice Hall, 1995.
- Turchin, Y., Gal, A., & Wasserkrug, S. (2009, July). Tuning complex event processing rules using the prediction-correction paradigm. In Proceedings of the Third ACM International Conference on Distributed Event-Based Systems (p. 10). ACM.
- UK Mi5 Threat levels, Available: https://www.mi5.gov.uk/threat-levels
- Von Luxburg, U. (2007). A tutorial on spectral clustering. Statistics and computing, 17(4), 395-416.
- VSTEP EyeObserve Camera Operator Trainer developed by VSTEP http://vstepsimulation.com/product/eyeobserve/.
- Vu, C.T., Beyah, R.A., Li, Y.: Composite Event Detection in Wireless Sensor Networks. In: Proceeding of the IEEE International Performance, Computing, and Communications Conference, pp. 264–271 (2007).
- Walzer, K., Groch, M., & Breddin, T. (2008, September). Time to the rescue-supporting temporal reasoning in the rete algorithm for complex event processing. In Database and Expert Systems Applications (pp. 635-642). Springer Berlin Heidelberg.
- Wang, F., Liu, S., & Liu, P. (2009). Complex RFID event processing. The VLDB Journal—The International Journal on Very Large Data Bases, 18(4), 913-931.
- Wang, M., Perera, C., Jayaraman, P. P., Zhang, M., Strazdins, P., & Ranjan, R. (2015). City Data Fusion: Sensor Data Fusion in the Internet of Things. arXiv preprint arXiv:1506.09118.
- Weinstein, C., Campbell, W., Delaney, B., & O'Leary, G. (2009, March). Modeling and detection techniques for counter-terror social network analysis and intent recognition. In Aerospace conference, 2009 IEEE (pp. 1-16). IEEE.
- Whitmore, A., Agarwal, A., & Da Xu, L. (2015). The Internet of Things—A survey of topics and trends. Information Systems Frontiers, 17(2), 261-274.
- Widder, A., Ammon, R. V., Schaeffer, P., & Wolff, C. (2007, June). Identification of suspicious, unknown event patterns in an event cloud. In Proceedings of the 2007 inaugural international conference on Distributed event-based systems (pp. 164-170). ACM.
- Wilkinson, P. (1986). Terrorism and the liberal state.
- Wu X., Zhu X., Wu G., Ding W. (2014), "Data Mining with Big Data". IEEE Trans. Knowl. Data Eng. 26(1), p 97-107.
- Wu, E., Diao, Y., & Rizvi, S. (2006, June). High-performance complex event processing over streams. In Proceedings of the 2006 ACM SIGMOD international conference on Management of data (pp. 407-418). ACM.
- Xue, A., Wang, W., & Zhang, M. (2011). Terrorist organization behavior prediction algorithm based on context subspace. In Advanced Data Mining and Applications (pp. 332-345). Springer Berlin Heidelberg.

- Yang, C. C., & Ng, T. D. (2008, June). Analyzing content development and visualizing social interactions in web forum. In Intelligence and Security Informatics, 2008. ISI 2008. IEEE International Conference on (pp. 25-30). IEEE.
- Yang, M., Kiang, M., Chen, H., & Li, Y. (2012). Artificial immune system for illicit content identification in social media. Journal of the American Society for Information Science and Technology, 63(2), 256-269.
- Yang, Y., Guan, X., & You, J. (2002, July). CLOPE: a fast and effective clustering algorithm for transactional data. In Proceedings of the eighth ACM SIGKDD international conference on Knowledge discovery and data mining (pp. 682-687). ACM.
- Zuech, R., Khoshgoftaar, T. M., & Wald, R. (2015). Intrusion detection and big heterogeneous data: A survey. Journal of Big Data, 2(1), 1-41.
- Zyda, M. (2005). From visual simulation to virtual reality to games. Computer, 38(9), 25-32.

Sitography

[S1] http://www.ibiscorp.com/Anti%20terrorism%20consulting.html

[S2] http://www.s2institute.com/

[S3] http://www.wrmi-llc.com/index.html

[S4] Asal, Victor, Amy Pate and Jonathan Wilkenfeld. 2008. Minorities at Risk Organizational Behavior Data and Codebook Version 9/2008 online: http://www.cidcm.umd.edu/mar/data.asp