Event Detection and Modelling for Security Application

Nur Farhan Binti Kahar

Submitted in partial fulfillment of the requirements of the Degree of Doctor of Philosophy

Supervisor: Prof. Ebroul Izquierdo

School of Electronic Engineering and Computer Science

Queen Mary University of London

United Kingdom

May 2019

STATEMENT OF ORIGINALITY

I, Nur Farhan Binti Kahar, confirm that the research included within this thesis is my own work or that where it has been carried out in collaboration with, or supported by others, that this is duly acknowledged below and my contribution indicated. Previously published material is also acknowledged below.

I attest that I have exercised reasonable care to ensure that the work is original, and does not to the best of my knowledge break any UK law, infringe any third party's copyright or other Intellectual Property Right, or contain any confidential material.

I accept that the College has the right to use plagiarism detection software to check the electronic version of the thesis.

I confirm that this thesis has not been previously submitted for the award of a degree by this or any other university.

The copyright of this thesis rests with the author and no quotation from it or information derived from it may be published without the prior written consent of the author.

Signature:

Date: 25^{h} May 2019

ABSTRACT

This thesis focuses on the design and implementation of a novel security domain surveillance system framework that incorporates multimodal information sources to assist the task of event detection from video and social media sources. The comprehensive framework consists of four modules including Data Source, Content Extraction, Parsing and Semantic Knowledge. The security domain ontology conceptual model is proposed for event representation and tailored in conformity with elementary aspects of event description. The adaptation of DOLCE foundational ontology promotes flexibility for heterogeneous ontologies to inter-operate. The proposed mapping method using eXtensible Stylesheet Language Transformation (XSLT) stylesheet approach is presented to allow ontology enrichment and instance population to be executed efficiently. The dataset for visual semantic analysis utilizes video footage of 2011 London Riots obtained from Scotland Yard. The concepts *person, face, police, car, fire, running, kicking* and *throwing* are chosen to be analysed. The visual semantic analysis results demonstrate succesful persons, actions and events detection in the video footage of riot events. For social semantic analysis, a collection of tweets from twitter channels that was actively reporting during the 2011 London Riots was compiled to create a Twitter corpus. The annotated data are mapped in the ontology based on six concepts: *token, location, organization, sentence, verb,* and *noun*. Several keywords related to the event that has been presented in the visual and social media sources are chosen to examine the correlation between both sources and to draw supplementary information regarding the event. The chosen keywords describe actions *running, throwing,* and *kicking;* activity *attack, smash* and *loot;* event *fire;* and location *Hackney* and *Croydon*. An experiment in respect to *concept-noun* relations are also been executed. The ontology-based visual and social media analysis yields a promising result in analysing long content surveillance videos and lengthy text corpus of social media user-generated content. Adopting ontology-based approach, the proposed novel security domain surveillance system framework enables a large amount of visual and social media data to be analysed systematically and automatically, and promotes a better method for event detection and understanding.

TABLE OF CONTENTS

LIST OF FIGURES

LIST OF TABLES

LIST OF ABBREVIATIONS

CCTV - Closed-circuit Television

BSIA - British Security Industry Authority

FOL - First Order Logic

DL - Description logic

HOG - Histograms of Oriented Gradients

ViPER-GT - Video Performance Evaluation Resource - Ground Truth

GATE - General Architecture for Text Engineering

XML - eXtensible Markup Language

OWL - Web Ontology Language

XSL - eXtensible Stylesheet Language

XSLT - eXtensible Stylesheet Language Transformation

RDF - The Resource Description Framework

NLP - Natural Language Processing

ANNIE - A Nearly-New Information Extraction

POS - Part-of-Speech

NE - Named Entities

JAPE - Java Annotation Patterns Engine

SAX - Simple API for XML

SWRL - Semantic Web Rule Language

SPARQL - SPARQL Protocol and RDF Query Language

DOLCE - Descriptive Ontology for Linguistic and Cognitive Engineering

RDFS - Research Description Framework Schema

CHAPTER 1

INTRODUCTION

Recent developments in multimedia technologies have led to the generation of vast quantities of multimedia data from a variety of sources and sensors. This trend has, in turn, originated a wave of research addressing automation of related information mining and understanding. However, there is still a significant gap in the automated understanding of very large volumes of raw, multimodal data capturing single real-world events through different sources including closed-circuit television (CCTV), mobile devices and social media. This problem becomes critical in a forensic context for key security applications. As a consequence, the need for an automated understanding of information for crime detection and prevention is of paramount relevance [1].

1.1. Problem Statement

Globally, hundreds of millions of CCTV cameras have been installed for surveillance purposes, producing enourmous quantities of big data in the form of video footage. In the United Kingdom, The British Security Industry Authority (BSIA) estimate that there are up to 5.9 million CCTV cameras with approximately 500, 000 surveillance cameras running in London alone [2] [3]. One of the biggest challenges when reviewing the video for investigations, is thus the sheer volume of video footage which may need to be examined. For example, typically it may take a trained officer or analyst using traditional methods (a notepad and the pause/rewind buttons) 1.5 to 2 hours to review just an hour of raw video footage [4]. This leads to large resource consumption and cost. The problem is particularly prominent in police forces where the issue is amplified by a spike in the number of video inputs through increased use of body-worn cameras and more publicly submitted videos.

Compounding the scale of the problem, social media has also recently been used for surveillance purposes through leveraging information shared by its users to gather updates about situations in various locations at different times. As of the first quarter of 2018, Twitter averaged 336 million monthly active users [5] and Facebook records over 2.19 billion monthly active users worldwide [6]. Thus with so much potential, this approach is also facing the same issues, i.e. the sheer scale of vast quantities of textual data produced by social media users hinders the process of ascertaining critical and useful information for investigations. Furthermore, the different format in visual and social media data discourages comprehensive event information analysis to be done simultaneously for both resources. The challenges of interpreting the data are compounded by increment in the volumes of data needing to be reviewed in situations such as riots, where wide scale public involvement across potentially large areas takes place. Therefore, related initiatives recognize the need for better software tools for helping officers to identify relevant events from multimodal resources especially in the security domain.

1.2. Proposed Solution and Approach

One of the trends that has become prevalent recently is leveraging the ontology-based approach to represent and formally specify the knowledge related to a domain [7]. Ontologies provide a common understanding of domains that can be communicated between people and application systems. An ontology-based system involves two different sub-systems, low-level and high-level processing. In low-level processing, the raw data is processed by low-level processing algorithms, i.e. algorithms usually generating feature descriptions, sets of symbolic descriptors which summarizes characteristics of data in a quantitative way. Alternatively, high-level processing is related to data interpretation and reasoning with data. High-level processing is usually built on top of the low-level processing algorithms, taking features descriptors as input and generating abstract, qualitative descriptions about the content of the data. High-level processing makes use of domain-level representation where events are typically subject to discussions and interpretations by humans and may be very complex, with a variety of aspects need to be considered.

The research work leading to this thesis proposes and implements a novel integrated framework to support the modelling and semantic reasoning of event information from multimodal resources. The aim of this research is to develop an automated surveillance system for event detection and understanding, utilizing visual data from CCTV footage and social media user-generated content. This synergised approach requires analysis of low-level visual and textual processing, event representation and high-level semantic reasoning. This semantic platform is innovative, as it integrates semantic and reasoning into a coherent operational software framework to support automated semantic analysis of surveillance domain datasets. The proposed system will help the forensic analyst to detect events from a vast collection of video and textual data using event-based semantics.

1.3. Objectives

1. To research, design and implement a novel security domain surveillance system framework to effectively assist the task of event detection from video and social media data sources. (See Chapter 3)
2. To build an event conceptual model for security domain ontology to support knowledge representation and semantic reasoning of multimedia data. (See Chapter 4)
3. To implement an automated and manual feature extraction approach to extract the visual content descriptions from video footage and textual user-generated content from social media platform for ontological analysis. (See Chapter 2, 3, 5 and 6)
4. To develop and implement the parsing process to execute inter-level data transformation between lower-level syntactic data and higher-level semantic concepts. (See Chapter 3 and Chapter 5)
5. To validate the proposed innovative framework by implementing rule-based semantic reasoning on conceptual knowledge and information retrieval from the domain ontology. (See Chapter 5 and Chapter 6)

1.4. Contribution of the Thesis

The contribution of this thesis lays in the development and implementation of a novel ontology-based security domain surveillance system which supports analysis of the vast collection of visual data as well as social media user-generated content to achieve high-level interpretation and understanding of events. The research work is described as the following:

1. Design and implementation of an ontology-based security domain surveillance framework that incorporates multimodal information sources, including visual information from CCTV footage and textual information from the social media platform. The comprehensive

framework consists of four modules which includes Data Source, Content Extraction, Parsing and Semantic Knowledge. These modules administrate data processing from data source level to semantic output. The functionalities and processes of each module are elaborated in Chapter 3.

2. The development of an event conceptual model for security domain ontology through an implementation of ontology development methodology and the conceptualization classification extension in accordance with six elementary aspects which underpin functional requirements of an event model. The event model is built upon the foundational ontology DOLCE and provides support for ontology integration and reuse (See Chapter 4).

3. To support visual analysis, implementation of automated object detection to detect person and face features, as well as manual feature annotations to extract additional salient features from video footage are introduced. These research work are first introduced in Chapter 2, Section 2.4 and later demonstrated in Chapter 5. The Histogram of Oriented Gradient (HOG) method is used for person detection and Haar feature-based cascade classifier is used for face detection. Manual video annotation is executed using Video Performance Evaluation Resource (ViPER) tool.

4. The implementation of text processing and analysis of social media user-generated content to extract important information shared by social media users during the event. This topic is initially detailed in Chapter 2, Section 2.4.5 and subsequently executed in Chapter 6. Text annotation is executed using General Architecture for Text Engineering (GATE) annotation tool.

5. A mapping method is proposed to bridge the gap between lower-level syntactic data of extracted features represented in Extensible Markup Language (XML) and higher-level semantic concepts expressed in Web

Ontology Language (OWL) using eXtensible Stylesheet Language Transformation (XSLT) stylesheet approach. The mapping method allows ontology enrichment and instance population to be executed efficiently for a better event representation. This task is first detailed in Chapter 3, Section 3.5 and later substantiated in Chapter 5, Section 5.3.

6. Construct semantic rules using Semantic Web Rule Language (SWRL) and execute queries using SPARQL Protocol and RDF Query Language (SPARQL) to perform rule-based semantic reasoning and knowledge retrieval for both visual and social media analysis. These tasks are demonstrated in Section 5.5 to 5.7 and Section 6.6, both in Chapter 5 and 6 respectively.
7. Evaluation of the proposed framework using CCTV footage and social media user-generated content of real riot events is presented in Chapter 5 and Chapter 6. The experimental result and analysis validate the proposed security domain surveillance system framework as well as distinguished its strength and weaknesses.

1.5. Thesis Outline

This thesis presents a detailed literature review, novel system development, analysis and implementation as follows:

Chapter 2 provides a literature review and theories on the related research topic, including knowledge representation, ontology engineering methodology, information extraction approaches, meta data mapping and semantic reasoning.

Chapter 3 presents a framework for automated media analysis in a security domain surveillance system which highlights four main modules; the data source, content extraction, parsing and semantic knowledge module. The purpose and task of every module and submodule are also presented and discussed.

Chapter 4 elaborates on security domain ontology engineering, which demonstrates the definition of the domain event model, development methodology and conceptualization classifications based on foundational ontology.

Chapter 5 demonstrates visual analysis framework validation and presents the experimental results and analysis on feature extractions, XML to OWL parsing, rules generation, semantic reasoning and queries to verify its efficiency.

Chapter 6 presents social media semantic analysis framework validation on a Twitter corpus of 2011 London Riots, which involves text annotation process, ontology population, rules generation, semantic reasoning and queries to extract useful information from social media posts.

Chapter 7 concludes with a summary of the main contributions throughout the thesis, plus research recommendations and future work.

1.6. Publications

1. **N. F. Kahar** and E. Izquierdo, "Ontology-based analysis of CCTV data," *7th Latin American Conference on Networked and Electronic Media (LACNEM 2017)*, Valparaiso, 2017, pp. 62-67.

2. F. Sobhani, **N. F. Kahar** and Q. Zhang, "An ontology framework for automated visual surveillance system," *2015 13th International Workshop on Content-Based Multimedia Indexing (CBMI)*, Prague, 2015, pp. 1-7.

CHAPTER 2

SURVEY OF RELATED RESEARCH

This chapter presents a literature review and theories on the related research topic as a foundation for the following chapters. The chapter begins with discussion on current issues in surveillance application, knowledge representation requirements, ontological engineering methodologies and implementation of foundational ontology to facilitate ontology development. Subsequently, object detection and language processing approaches for low-level analysis and metadata mapping for inter-level data transformation is presented. Finally, semantic reasoning and knowledge retrieval are implemented for high-level processing.

2.1. Surveillance Application

Today, CCTV surveillance has become a prominent aspect of day-to-day life as an approach to crime prevention, traffic monitoring, and home security. According to the IHS, there were 245 million professionally installed video surveillance cameras active and operational globally in 2014 [8] increasing to 350 million in 2016 [9]. The British Security Industry Authority (BSIA) estimated that there are up to 5.9 million CCTV cameras in the UK alone [2], with approximately 500,000 in London [3].

Current state-of-the-art surveillance systems are based either on the statistical analysis of image features, or on the hard-coded interpretation of object identification [10]. The complexity range of situations is very wide, ranging from the mere detection of movement that sets off an alarm, to an integral control system that monitors the scene with different sensors, diagnoses the situation and plans a series of consistent actions [11]. However, although there is a huge amount of CCTV data gathered, its problematic that there is no efficient way to analyse it. The enormous collection burden and efficient analysis of data can impact the timely process of justice, resulting in a major drawback especially for a security-related domain.

In addition to the visual surveillance approach, recent years have also witnessed a rapid increase in social media usage, which has also become an important source for reporting real-world events [12]. This trend has also led to a substantial body of research in the generic field of visual information retrieval with applications to several domains of science and technology [13], [14], [15], [16], [17]. The substantial amount of useful information produced by the public's collective intelligence is highly beneficial for surveillance purpose. Shared information in the form of text posts, photographs, and videos gives valuable multi-perspective information that can communicate a coherent story about an event in real time. The large volume of information shared by social media

users facilitates event detection and understanding which is ultimately useful for investigations by security forces. As of the first quarter of 2018, the social networks, Twitter averaged at 336 million monthly active users [5] and Facebook records over 2.19 billion monthly active users worldwide [6]. Thus enormous data is being generated at a very rapid rate.

Research on event detection using social media data has been carried out for multiple purposes using various algorithms and approaches. The current literature includes emergency management during a large-scale event [18], real-time identification of small-scale incidents [19] and emergency situation awareness during disaster and crisis [20]. By harvesting additional information about incidents from social media, these studies contribute to enhancing situation awareness through early incident indicator identification, the exploration impact, and incidents' evolution monitoring. In [21], real-time earthquake event detection is performed through the investigation of the real-time interaction of events in Twitter and consequently, an implementation of algorithms to monitor tweets and to detect earthquakes. Abnormal topics and event detection within various social media data sources, such as Twitter, Flickr and YouTube are presented in [22].

However, to the best of our knowledge, to date, there have been no attempts to perform semantic analysis of visual-social media content for surveillance applications. Common approaches often only focus on single sources of information. For instance, some works either do not consider additional, external information beyond photos and videos at all, or only do so to a limited degree [13]. In relation to semantic analysis implementation, previous research on ontology-based approach for social media analysis focus in domains such as crisis management [23], [24] criminal digital evidence [25], sentiment analysis [26], business [27] and urban planning [28]. On the other hand, research on automated video analysis using ontologies has been carried out in [29], [30],

[31], [32] and produces promising results. There is thus a growing need for an automated event detection and understanding which focuses on surveillance applications leveraging a synergy of both visual and social media information sources.

2.2. Knowledge Representation

Knowledge representation is the field of artificial intelligence that focuses on the formalism design [33], [34] where the knowledge about a specific domain is expressed epistemologically and computationally with an objective of solving complex problems. Knowledge representation makes complex software easier to define and maintain than procedural code and can be used in expert systems. Knowledge representation goes hand in hand with automated reasoning because one of the primary purposes of explicitly representing knowledge is to be able to reason about that knowledge, to make inferences, assert new knowledge, etc. Virtually all knowledge representation languages have a reasoning or inference engine as part of the system.

2.2.1 Requirements of a Knowledge Representation

Discussions on properties of a good knowledge representation system for any domain in [35] clearly stated that the following properties should be possessed:

1. Representational Adequacy
    - the ability to represent all the different kinds of knowledge that might be needed in that domain.
2. Inferential Adequacy
    - the ability to manipulate the representational structures to derive new structures (corresponding to new knowledge) from existing structures.

3. Inferential Efficiency
    - the ability to incorporate additional information into the knowledge structure which can be used to focus the attention of the inference mechanisms in the most promising directions.
4. Acquisitional Efficiency
    - the ability to acquire new information easily. Ideally, the agent should be able to control its own knowledge acquisition.

Knowledge representation can be broken down into four fundamental components for analysis purposes. The first component is the *lexical part* that determines symbols or words that are used in the representation's vocabulary. Next is the *structural or syntactic part*, that describes the constraints on how the symbols can be arranged. The *semantic part* establishes a way of associating real-world meanings with the representations and finally, the *procedural part* is the one that specifies the access procedures that enables ways of creating and modifying representations and answering questions using them.

2.2.2 Logical Representation

In mathematical logic, propositional logic is the logic whose formulae are made of atomic propositions like *A*, *B*, having always one of the values true or false (truth values), and logical connectives like negation $(\neg A)$, and $(A \wedge B)$, or $(A \vee B)$ and implication $(A \rightarrow B)$ [36]. Propositional logic is *sound* (only deriving correct results), *complete* (able to derive any logically valid formula) and *decidable* (the algorithms for deciding whether a formula is valid end in finite time). Predicate logic is the logic which adds predicates (like *P(x,y)*) which represent *relations*, i.e. produce true or false for a combination of values of the terms *x* and *y*; quantifiers: *existential* ∃ ("there exists") and *universal* ∀ ("for all"); and terms made of variables and functions, like *f(x)*, *g(y,z)*. Thus, predicate logic can form formulas like $\forall x \exists y(P(x) \rightarrow Q(f(y)))$.

First order predicate logic [37] is the logic where the quantifiers can range only over elements of sets. In higher order logics, quantifiers can range over sets, functions and other objects. So, for example, the sentence that "every set of real numbers has a minimum" cannot be expressed in first order logic, because the quantification ranges over sets, not set elements. First order predicate logic is *sound* and *complete*; however, it is *not decidable*. It is semi-decidable, i.e. valid formulas can be proven, but non-valid formulas may need infinite time to construct a counter-example of infinite size. There are known algorithms that can prove valid theorems in first order predicate logic, namely the tableaux algorithm, however, if the theorem is not valid, the algorithm may not end in finite time.

The ultimate knowledge representation formalism in terms of expressive power and compactness is First Order Logic (FOL) [38]. There is no more powerful formalism than that used by mathematicians to define general propositions about the world. However, FOL has two drawbacks as a knowledge representation formalism: ease of use and practicality of implementation. First order logic can be intimidating even for many software developers. Languages which do not have the complete formal power of FOL can still provide close to the same expressive power with a user interface that is more practical for the average developer to understand. The issue of practicality of implementation is that FOL in some ways is too expressive. With FOL it is possible to create statements (e.g. quantification over infinite sets) that would cause a system to never terminate if it attempted to verify them. Thus, a subset of FOL can be both easier to use and more practical to implement. This was a driving motivation behind rule-based expert systems [39]. *IF-THEN* rules provide a subset of FOL but a very useful one that is also very intuitive. The history of most of the early artificial intelligence knowledge representation formalisms; from databases to semantic nets to theorem provers and production systems can be viewed as various design decisions on whether to emphasize expressive power or computability and efficiency.

Description Logics (DLs) [40], [41], [42] are a family of knowledge representation languages that can be used to represent the knowledge of an application domain in a structured and formally well-understood way. The name description logics is motivated by the fact that, on the one hand, the important notions of the domain are described by concept descriptions, i.e., expressions that are built from atomic concepts (unary predicates) and atomic roles (binary predicates) using the concept and role constructors provided by the particular DL. On the other hand, DLs differ from their predecessors, such as semantic networks and frames, in that they are equipped with a formal, logic-based semantics.

2.2.3 Ontology

An ontology is a “formal specification of a shared conceptualization” [43]. Ontology is a conceptual model in a domain which is used to represent the concepts and relationship through them, which contains a description of the specific domain [33]. One of the main advantages of using ontologies is their way to represent and share knowledge by using a common vocabulary. As providers of a format for exchanging knowledge, ontology promotes interoperability, knowledge reuse, and information integration with automatic validation. They separate declarative and procedural knowledge, making the modularity of the knowledge base easier [44]. Ontologies allow information to become not only human but also machine readable and processable. Multimedia ontologies have been designed in order to serve one or more of the following tasks:

- **Annotation** – tagging or labelling multimedia content
- **Analysis** – ontology-driven semantic analysis of multimedia content
- **Retrieval** – context-based image retrieval
- **Personalization** – recommendation and filtering of multimedia content based on user preferences

- **Algorithms and processes control** – modelling multimedia procedures and processes
- **Reasoning** – personalization and retrieval for creating autonomous content applications.

An ontology consists of four main components to represent a domain. They are:

- ***Concept*** represents a set of entities within a domain
- ***Relation*** specifies the interaction among concepts
- ***Instance*** indicates the concrete example of concepts within the domain
- ***Axioms*** denote a statement that is always true

2.3. Ontological Engineering

Constructing a domain model, or ontology is an important step in the development of knowledge-based systems. The advantages of such domain models have been widely canvassed, and include enabling the sharing of knowledge, the re-use of knowledge, and the better engineering of knowledge-based systems with respect to acquisition, verification, and maintenance [45].

At present, the construction of ontologies is very much an art rather than a science. The movement from the ontological art to the ontological engineering lead researchers to propose different methodologies, in order to develop ontologies for different purposes in different fields. In the context of ontology development methodologies, a considerable number of surveys have been conducted in the literature [46]. An ontology methodology describes the necessary activities that should be carried out, how to carry out every activity, the order of these activities and the required techniques that should be used to implement and maintain the ontology [47].

2.3.1 Ontology Engineering Methodology

Considerable effort has been channelled into the aim of proposing methodologies for ontology development in the literature thus far. The most widely known methodologies are overviewed below.

Uschold and King [48] proposed a methodology for building ontologies based on four primitive activities: identify the purpose, building the ontology, evaluation, and documentation. The building activity includes three sub-activities: ontology capture, ontology coding and integrating existing ontologies. In addition, the authors of this study add that these activities should include a set of techniques, methods, principles, and guidelines for each stage, as well as indicating what relationships exist between the stages. However, the methodologies for carrying out the evaluation activities are not covered [49].

Similarly, Gruninger and Fox [50] proposed another methodology called TOVE (TOronto Virtual Enterprise). TOVE proposes six activities starting from motivation scenario, informal competency question, terminology, formal competency question, axiom and ending with completeness theorem. The approach includes defining an ontology's requirements, defining the terminology of the ontology, specifying the definitions and constraints on the terminology and finally testing the competency of the ontology by providing completeness theorems with respect to the competency questions. In TOVE, building the ontology is based on competency questions [51]. However, some activities such as knowledge acquisition, documentation, and maintenance are not explicitly stated in TOVE [46].

METHONTOLOGY is an alternative methodology proposed by Fernández et al. [52], which is considered to be a complete methodology for ontology building [51]. The authors propose to have reduced the existing gap between ontological

art and ontological engineering by identifying a set of activities to be conducted during the ontology development process, proposing the evolving prototype of ontology life cycle and defining METHONTOLOGY, a well-structured methodology to build ontologies from scratch. In METHONTOLOGY, building ontology from scratch is composed of seven activities, which are specification, knowledge acquisition, conceptualization, integration, implementation, evaluation, and documentation. This methodology provides a clear guidance in which the process of carrying out every activity is defined clearly [46]. Furthermore, the ontology life cycle proposed in METHONTOLOGY provides an accurate description of every activity [53]. The definition of the ontology development process is based on IEEE Standard 1074-1995 [54].

The methodology for building ontology in public administration from scratch is proposed by [53]. The development of this methodology is based on [52] and [50]. This methodology is composed of three main sub-processes specification, concretization, and implementation. The orders of these sub-processes are very important, since the output of each subprocess will be used as an input for the next one. This methodology considers the graphical representational (in the subprocess concretization), which is not explicitly considered in the aforementioned methodologies.

Similarly, De Nicola et al. [51] proposed another methodology for building ontology based on the software engineering Unified Process (UP), a highly scalable and customizable methodology. The new methodology called UPON stands for UP for Ontology. Following UP approach, there are cycles, phases, iterations and workflows in UPON. Each cycle consists of four phases (inception, elaboration, construction, and transition) and each phase is further subdivided into iterations. During each iteration, five workflows take place which is requirements, analysis, design, implementation, and test. It is noteworthy to point out that, UPON provides a clear and accurate description for each of the

workflows that are involved in the process of building ontology based on UP. Based on the aforementioned analysed methodologies, Table 2.1 summarizes the activities that involved in the ontology development life cycle [55]. The common ontology development life cycle from ontology engineering perspective involves around the following phases:

Table 2.1: Ontology engineering phases

Phase		**Activities**
Requirement Analysis	Specification	Identifying ontology specification [56]: • The purpose of the ontology • The scope of the ontology • Target users of the ontology • Ontology usage scenarios • User requirements • Ontology requirements - equipment and software
	Knowledge Acquisition	Acquiring informal information related to knowledge and problem-solving process of subject matter experts using observation. Document analysis and structuring techniques.
Development	Conceptualization	Developing knowledge representation in a semi-formal format using graphical representation
	Formalization	Changing the semi-formal knowledge representation to formal knowledge representation
	Integration	Identifying any appropriate existing ontology that can be integrated into the ontology being developed

Implementation	Implementation	Transforming human-readable representation into a machine-readible representation
Evaluation Maintenance	Evaluation and Maintenance	Evaluating and assessing the developed ontology in meeting the required specifications. Identifying individuals to update and maintain the developed ontology.
Documentation	Documentation	Includes writing the necessary documentation to facilitate the use, reuse, and maintenance of the ontology, as well as, for enhancing the clarity of the ontology.

2.3.2 Foundational Ontology

A foundational ontology, sometimes also called ‘upper-level ontology’, defines a range of top-level domain-independent ontological categories, which form a general foundation for more elaborated domain-specific ontologies [57]. Foundational ontologies are ultimately devoted to facilitate mutual understanding and inter-operability among people and machines. This includes understanding the reasons for non-interoperability, which may in some cases be much more important than implementing an integrated system relying on a generic shared “semantics”. The role and nature of foundational ontology building require more painful human labour, yet immense benefit can be gained from the results and methodologies of disciplines such as philosophy, linguistics, and cognitive science.

The advantage of using a foundational ontology is it facilitates ontology development since one does not have to reinvent the wheel concerning basic categories and relations during the development process [58]. The foundational ontology also serves as modelling guidance for ontology development and

improves overall ontology quality and interoperability. Existing Upper Ontologies includes UFO (Unified Foundational Ontology), BFO (Basic Formal Ontology), DOLCE (Descriptive Ontology for Linguistic and Cognitive Engineering), SUMO (Suggested Upper Merged Ontology), YOMATO (Yet Another More Advanced Top-level Ontology), GFO (General Formal Ontology), PROTON (PROTo Ontology) and Cyc. Table 2.2 shows the comparison between foundational ontological commitments as discussed in [59].

Table 2.2: Comparison of ontological commitments

Foundational Ontology	Language(s)	Modularity	Applications
DOLCE	First Order Logic, KIF, OWL	Not divided into modules	Multilingual information retrieval, web-based systems, and services, e-learning
BFO	OWL	SNAP and SPAN modules	Mainly in the biomedical domain
GFO	First Order Logic and KIF, OWL	Abstract top level, abstract core level, basic level	Mainly in the biomedical domain
SUMO	SUO-KIF, OWL	Divided into SUMO itself, MILO, and domain ontologies	Linguistics, representation, reasoning
Cyc	CycL, OWL	“Microtheory” modules	Natural language processing, network risk assessment, terrorism management
PROTON	OWL Lite	Three levels including four modules	Semantic annotation within the KIM platform, knowledge management systems in legal and telecommunications domain, media research and analysis, research intelligence, Business Data Ontology for Semantic Web Services.

As reflected by its acronym, DOLCE has a clear cognitive bias, in the sense that it aims at capturing the ontological categories underlying natural language and human common sense. DOLCE is an ontology of particulars, in the sense that its domain of discourse is restricted to them. The fundamental ontological distinction between universals and particulars can be characterized by means of the primitive relation of instantiation: particulars are entities that cannot have instances; universals are entities that can have instances. In linguistic, ‘proper nouns’ are normally considered to refer to particulars, while ‘common nouns’ to universals. For example, ‘Varenne’, the Italian racehorse, is an instance of ‘horse’, but it cannot be instantiated itself [60]. DOLCE’s abstract concepts are aimed at generalizing the set of concepts that may be encountered in different domains [61]. The taxonomy of the most basic categories of particulars assumed in DOLCE is depicted in Figure 2.1. Implementation of DOLCE’s basic categories in security domain ontology modeling are presented comprehensively in Chapter 4, Security Domain Ontology.

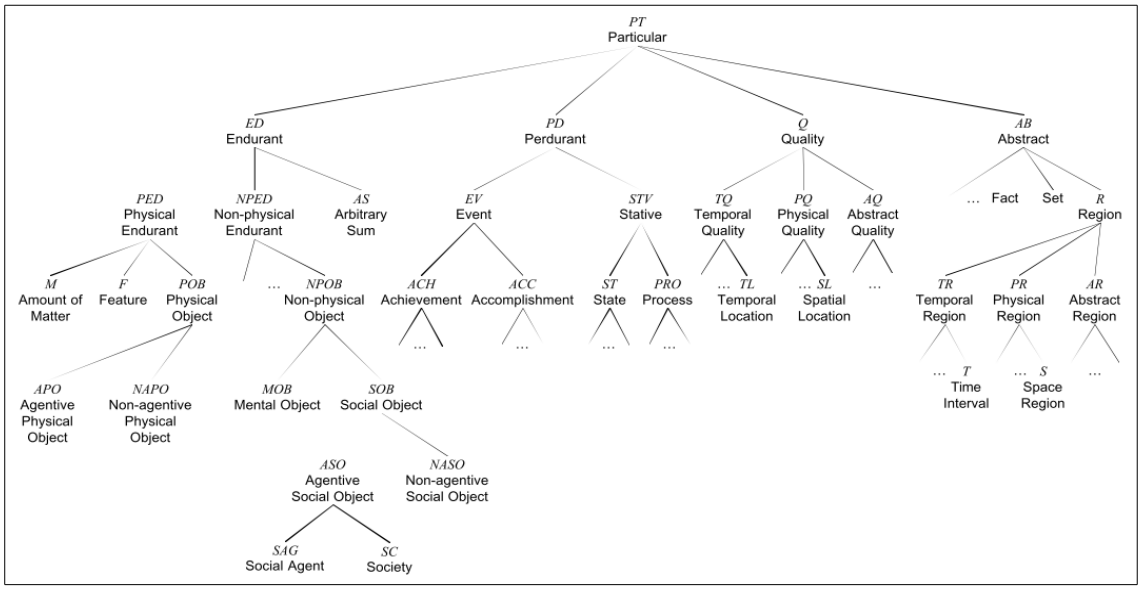


Figure 2.1: Taxonomy of DOLCE basic categories

2.4. Syntactic Information Extraction

The development of high-powered computers, the availability of high quality and inexpensive video cameras, and the increasing need for automated visual

analysis has induced a great deal of interest in object detection and tracking of objects. Detection of moving objects in video streams is the first relevant step of information extraction in many computer vision applications, including video surveillance, people tracking, traffic monitoring, and semantic annotation of videos [62].

2.4.1 Object Detection Approaches

There are many methods and approaches to detect objects reported in the literature. Methods such as Haar-wavelets as used by [63] for face detection are very fast but not that robust. Models based on Local Binary Patterns (LBP) are more suitable for face detection. More recent algorithms are Histogram of Oriented Gradient (HOG-features) [64] are far more robust which is more suited for complex objects such as pedestrians. For an even better accuracy, detectors based on Integral Channel Features (ICF) by [65] which combines the integral images for speed with multiple channels (both colour and gradient features) can be used. Another possibility is to use local features such as SURF, SIFT or FAST, incorporated with a bag of words approach. This approach has been used with success to recognize signs in a building, paintings and such like, while walking around with a camera. This approach will not result in a bounding box around the object, instead generating a number of matches between an object library and the object to classify. Other approaches for object detection [62] are shown in Figure 2.2 and described below.

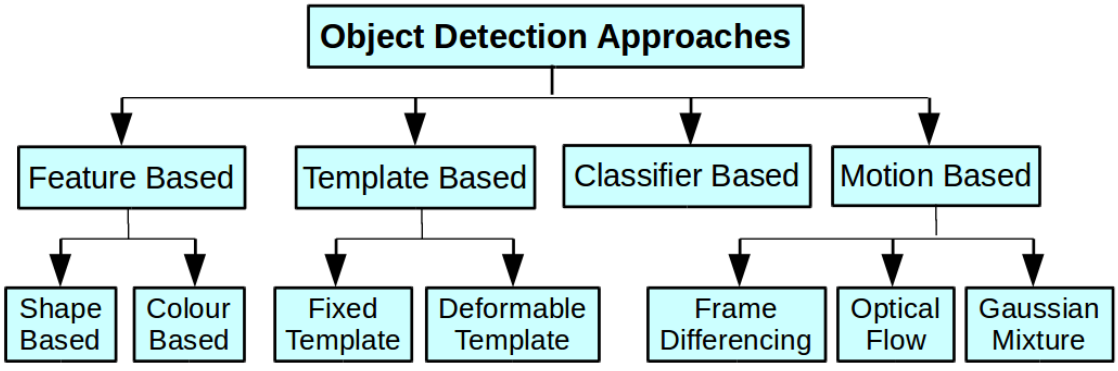


Figure 2.2: Object detection approaches

Feature Based Object Detection

In feature based object detection, standardization of image features is important. One or more features are extracted, and the objects of interest are modelled in terms of these features. Features may be shape, size or the colour of objects.

Template Based Object Detection

If a template describing a specific object is available, object detection becomes a process of matching features between the template and the image sequence under analysis. There are two types of object template matching, fixed and deformable template matching.

Motion Based Object Detection

Viola, Jones, and Snow [66] use motion information for detection of pedestrian. They use different motion filters for effective detection of pedestrians. A large variety of motion detection algorithms had been proposed such as frame differencing, optical flow and Gaussian mixture.

Classifier Based Object Detection

In classifier-based object detection, the separation of the video objects from the background is treated as a classification problem. A classifier with a set of parameters was built up based on the knowledge of the interest object. For complex objects, multiple classifiers needed to be integrated, which was called cascade classifiers or boosted classifiers. The basic idea of these cascade classifiers is that several weak classifiers are used to cover different features of the object and combined to reach a better classification globally. The limitation of this method is, more object features need to be embedded to train the object model under different environment and light conditions.

2.4.2 Object Detection using Haar-based Cascade Classifier

Object Detection using Haar feature-based cascade classifiers is an effective

object detection method proposed by Paul Viola and Michael Jones in their 2001 paper, “Rapid Object Detection using Boosted Cascade of Simple Features”. They propose a machine learning based approach where a cascade function is trained from thousands of positive and negative images before it can be used to detect objects in other images.

2.4.2.1 Features

Viola and Jones face detection procedure classifies images based on the value of simple features [67]. Haar-like features shown in Figure 2.3 are used for this purpose. Viola and Jones proposed a few motivations for using features rather than the pixels directly. A primary factor is that features can act to encode ad-hoc domain knowledge that is difficult to learn using a finite quantity of training data. The feature-based system also operates much faster than a pixel-based system.

Three kinds of features are being used to perform the image classification. The value of *two-rectangle feature* is the difference between the sum of the pixels within two rectangular regions. The regions have the same size and shape and are horizontally or vertically adjacent (See Figure 2.3). A *three-rectangle feature* computes the sum within two outside rectangles subtracted from the sum in a center rectangle. Finally a *four-rectangle features* computes the difference between diagonal pairs of rectangles.

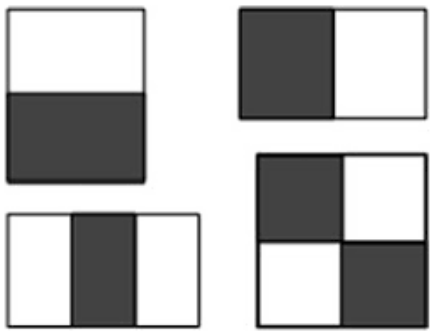


Figure 2.3: Haar-like features

2.4.2.2 Integral Image

In order to compute these features very rapidly at many scales , an intermediate representation for the image which is called integral image are being introduced. The integral image can be computed from an image using a few operations per pixel. Once computed, any one of these Haar-like features can be computed at any scale or location in constant time.

The integral image at location *x,y* contains the sum of the pixels above and to the left of *x,y*, inclusive:

$$ii(x, y) = \sum_{x' \leq x, y' \leq y} i(x', y'),$$

where *ii(x,y)* is the integral image and *i(x,y)* is the original image. Refer to Figure 2.4. Using the following pair of recurrences:

$$s(x, y) = s(x, y - 1) + i(x, y)$$
$$ii(x, y) = ii(x - 1, y) + s(x, y)$$

(where *s(x,y)* is the cumulative row sum, *s(x, -1)* = 0, and *ii(-1, y)* = 0) the integral image can be computed in one pass over the original image.

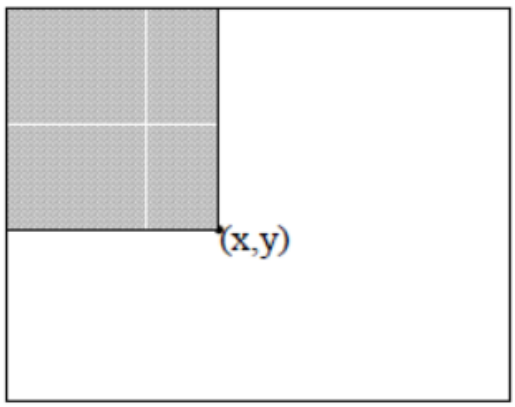


Figure 2.4: Integral image

Using the integral image, any rectangular sum can be computed in four array references. Refering to Figure 2.5, the value of integral image at location 1 is the sum of the pixels in rectangle A. The value at location 2 is A+B, at location 3 is A+C, and at location 4 is A+B+C+D. The sum within D can be computed as 4+1-(2+3). Since the two-rectangle Haar-like features as shown in Figure 2.3 involve adjacent rectangular sums, they can be computed in six array references, eight in the case of three-rectangle features, and nine for four-rectangle features.

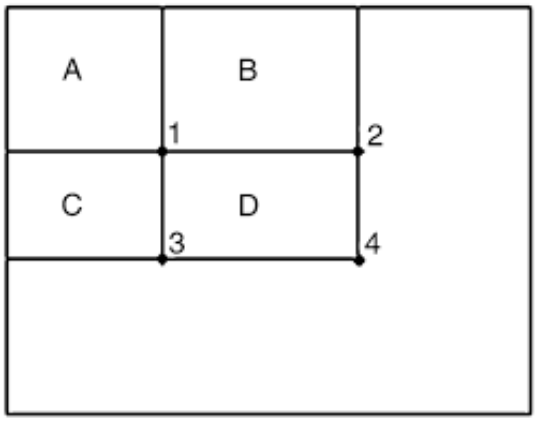


Figure 2.5: Integral image computation

Given a feature set and a training set of positive and negative images, any number of machine learning approaches could be used to learn a classification function. However, despite having an efficient feature computation technique, computing hundreds of thousands of rectangles features associated with each image sub-window is still prohibitively expensive. Therefore, in order to select critical visual features which could be combined to form an effective classifier, a variant of AdaBoost is used both to select the features and to train the classifier.

2.4.2.3 Learning Classification Functions using AdaBoost

In its original form, the AdaBoost learning algorithm is used to boost the classification performance of a simple learning algorithm. It works by combining

a collection of weak classifications functions to form a stronger classifier. In the language of boosting the simple learning algorithm is called weak learner. For example, the perceptron learning algorithm searches over the set of possible perceptros and returns the perceptron with the lowest classification error. The learner is called weak because we do not expect even the best classification function to classify the training data well. In order for the weak learner to be boosted, it is called upon to solve a sequence of learning problems. After the first round of learning, the examples are re-weighted in order to emphasize those which were incorrectly classified by the previous weak classifier. The final strong classifier takes the form of a perceptron, a weighted combination of weak classifiers followed by a threshold.

AdaBoost is an aggressive mechanism for selecting a small set of good classification functions which nevertheless have significant variety. Drawing an analogy between weak classifiers and features, AdaBoost is an effective procedure for searching out a small number of good “features” which nevertheless have significant variety. One practical method for completing this analogy is to restrict the weak learner to the set of classification functions each of which depend on a single feature. In support of this goal, the weak learning algorithm is designed to select the single rectangle feature which best separates the positive and negative examples. For each feature, the weak learner determines the optimal threshold classification function, such that the minimum number of examples are misclassified. A weak classifier *(h(x, f, p, θ))* thus consists of a feature *(f)*, a threshold *(θ)* and a polarity *(p)* indicating the direction of the inequality:

$$h(x, f, p, \theta) = \begin{cases} 1 & \text{if } pf(x) < p\theta \\ 0 & \text{otherwise} \end{cases}$$

Here, *x* is a 24 x 24 pixel sub-window of an image. AdaBoost learning algorithm is presented in detail in [63].

2.4.2.4 The Attentional Cascade

Viola and Jones also propose an algorithm for constructing a cascade of classifiers which achieves increased detection performance while radically reducing computation time [67]. The key insight of this cascade of classifiers is that smaller, more efficient boosted classifiers can be constructed which reject many of the negative sub-windows while detecting almost all positive instances. Simpler classifiers are used to reject the majority of sub-windows before more complex classifiers are called upon to achieve low false positive rates.

Stages in the cascade are constructed by training classifiers using AdaBoost. Starting with a two-feature strong classifier, an effective face filter can be obtained by adjusting the strong classifier threshold to minimize false negatives.

The initial AdaBoost threshold, $\frac{1}{2}\sum_{t=1}^{T}\alpha_t$, is designed to yield a low error rate on the training data. A lower threshold yields higher detection rates and higher false positive rates. Based on performance measured using a validation training set, the two-feature classifier can be adjusted to detect 100% of the faces with a false positive rate of 50%.

The overall form of the detection process is that of a degenerate decision tree, or “cascade” (see Figure 2.6). A positive result from the first classifier triggers the evaluation of a second classifier which has also been adjusted to achieve very high detection rates. A positive result from the second classifier triggers the third classifier, and so on. A negative outcome at any point leads to the immediate rejection of the sub-window. The structure of the cascade reflects the fact that within any single image an overwhelming majority of sub-windows are negative. As such, the cascade attempts to reject as many negatives as possible at the earliest stage possible. While a positive instance will trigger the evaluation of every classifier in the cascade, this is an exceedingly rare event.

Much like a decision tree, subsequent classifiers are trained using those examples which pass through all the previous stages. As a result, the second classifier faces a more difficult task than the first. The examples which make it through the first stage are “harder” than typical examples. The more difficult examples faced by deeper classifiers push the entire receiver operating characteristic (ROC) curve downward. At a given detection rate, deeper classifiers have correspondingly higher false positive rates.

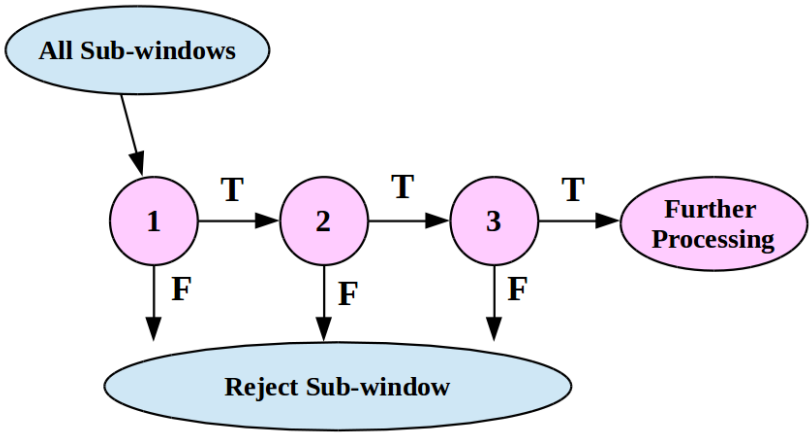


Figure 2.6: Schematic depiction of the detection cascade

2.4.3 Person Detection using Histograms of Oriented Gradients

Histograms of Oriented Gradients (HOG) is a type of “feature descriptor”. The intent of a feature descriptor is to generalize the object in such a way that the same object or person produces as close as possible to the same feature descriptor when viewed under different conditions. This makes the classification task easier. The HOG person detector was introduced by Dalal and Triggs in their paper “Histograms of Oriented Gradients for Human Detection” [64].

The HOG person detector uses a “global” feature to describe a person rather than a collection of “local” features. Thus, the entire person is represented by a single feature vector, as opposed to many feature vectors representing smaller parts of the person. The authors also trained a Support Vector Machine (SVM)

to recognize HOG descriptors of people. The HOG person detector uses a sliding detection window which is moved around the image. At each position of the detector window, a HOG descriptor is computed for the detection window. This descriptor is then shown to the trained SVM, which classifies it as either “person” or “not a person”.

2.4.4 Video Annotation using ViPER-GT

The Video Performance Evaluation Resource Ground Truth (ViPER-GT) toolkit allows annotation of a video with metadata, mainly for use as ground truth for performance evaluation [68]. This includes information describing the file, such as date of filming and keywords about its content. It also includes concrete features, such as scene breaks and bounding boxes around people. This can be used for any number of purposes. In this study, ViPER-GT is used to support a media database application, for instance, to track the movement of a person in a video. ViPER-GT can be used to go through the video frame by frame and mark up the movement by hand. ViPER-GT tool lets the user define boxes around people and its architecture allows integration with other tools. ViPER-GT is designed for editing visual annotation, such as rectangles denoting locations of people on screen. Shapes include points, bounding boxes and oriented rectangles, ellipses, polygons and circles, and annotation types without a visual element, including text strings, numbers, and Boolean values are also supported.

Data elements are combined together into objects called descriptors. This allows a person type to be defined, which has a text string (the person's name), a bounding box (their location in the frame), and any other number of attributes. Descriptors usually refer to a single object, event or other things in the file that is worthy of evaluation, but they may also have more abstract purposes, such as indicating keyframes. In addition to these types, all files have a single descriptor that gives metadata about the media file as a whole, including frame

rate, file name, image size in pixels, and an optional comment. ViPER-GT also maintains a set of descriptors associated to various source media files. The user can have one annotation file that describes several different media files, although it is often useful to have a one-to-one mapping of a media file to annotation file. The ViPER toolkit uses a simple data format (XGTF), to describe video content [69].

2.4.5 Language Processing using GATE

General Architecture for Text Engineering (GATE) is an infrastructure for developing and deploying software components that process human language [70]. Since one of the module in this study is focusing on analysing textual data from social media platform (Twitter), GATE is implemented to process Twitter corpus. In general, the core functions of GATE include:

- modelling and persistence of specialised data structures
- measurement, evaluation, benchmarking
- visualisation and editing of annotations, ontologies, parse trees, etc.
- a finite state transduction language for rapid prototyping and efficient implementation of shallow analysis methods
- extraction of training instances for machine learning
- pluggable machine learning implementations

On top of the core functions, GATE includes components for diverse language processing tasks, e.g. parsers, morphology, tagging, Information Retrieval tools, Information Extraction components for various languages, and many others. GATE Developer and Embedded are supplied with an Information Extraction system (ANNIE) which has been adapted and evaluated very widely. ANNIE is often used to create RDF or OWL metadata for unstructured content (semantic annotation).

GATE as an architecture suggests that the elements of software systems that process natural language can usefully be broken down into various types of components, known as resources. Components are reusable software chunks with well-defined interfaces and are a popular architectural form, used in Sun's Java Beans and Microsoft's .Net, for example. GATE components are specialised types of Java Bean, and come in three flavours:

- Language Resources (LRs) represent entities such as lexicons, corpora or ontologies;
- Processing Resources (PRs) represent entities that are primarily algorithmic, such as parsers, generators or ngram modellers;
- Visual Resources (VRs) represent visualisation and editing components that participate in GUIs.

2.5. Meta Data Mapping

Semantic web (OWL/RDF) worlds and eXtensible Markup Language (XML) [71] [72] have different data models, different semantics and use different query languages to access them. XML covers the syntactic level but lacks support for efficient sharing of conceptualizations. The Web Ontology Language (OWL) [73] [74] in turn, supports the representation of domain knowledge using classes, properties and instances for the use in a distributed environment as the World Wide Web. Therefore, it is crucial to develop tools and methodologies that will enable bridging the gap between them.

2.5.1 Formats and Languages

2.5.1.1 XML

XML is a simple, very flexible text format derived from SGML (ISO 8879) [75]. It was designed to flexibly structure information using markup. XML is a suitable language for exchanging a wide variety of data on the Web [76]. An XML

document is valid if it respects the grammar defined in a schema. The purpose of a schema is to define a class of XML documents [77]. In addition, eXtensible Stylesheet Language (XSL) is a specific language for defining style sheets associated with an XML document. An XSL style sheet is a file that describes how to transform a specific kind of XML document to another format. XSL includes three languages, eXtensible Stylesheet Language Transformation (XSLT), XPath, and XSL-FO. XSLT is used to transform XML documents using stylesheets containing rules called 'template rules'. XSLT uses XPath for designating a part of an XML tree [77]. XPath is a non-XML language used to address nodes in an XML document. Thus XPath is a query language commonly used in XSLT to specify paths in XML documents [78].

2.5.1.2 RDF and OWL

The Resource Description Framework (RDF) [71] [79] and Web Ontology Language (OWL) [73] [80] [81] are XML-based Semantic Web languages. These languages include a strong semantic definition which puts these languages at a higher level regarding usage of the XML language. XML was created to structure, store and exchange data between processes. The RDF language is a graph-based model that aims to describe Web resources formally and their metadata, such as the title and publication date of a web page. It is considered as a basic language for the Semantic Web. An RDF triple encodes a statement that is a simple logical expression or assertion about the world.

The OWL language is used to specify ontologies that are intended for publication and sharing on the Web with a higher level of logical expressivity with regards to the RDF language. OWL 1.0 consists of three sublanguages (OWL-Lite, OWL-DL,OWL-Full) of increasing expression. Each one is employed for specific users and requirements. In addition, each language is an extension of its simpler predecessor regarding the semantic richness. The OWL-Lite language is the simplest; being less expressive, it meets the requirements for a

classification hierarchy and functionality constraints for relationships. The OWL-DL language has semantic expressivity of the Description Logics. It is characterized by the completeness of the calculation and the decidability of the reasoning system. The OWL 2.0 language is actually the OWL-DL 2 language [82]. OWL-Full is characterized by the maximum expressiveness, however it cannot guarantee the completeness and decidability of calculations (which is the reason why it was not adopted by the Semantic Web community).

2.5.2 XML Data to OWL Ontologies Transformation

There are different approaches for transforming XML documents into OWL ontologies. They can be grouped into two classes according to the scheme from which the ontology is generated. A comparative study has been made between the approaches of each class and between classes themselves by [77]. The first approach called the 'instance approach' can generate an ontology semi-automatically and permits the automatic creation of an ontology or the enrichment of an existing ontology with the new content mapped. The second approach called the 'validation approach' mostly generate an ontology from an XSD or a DTD schema and is fully automated. The comparison between a family of 'validation approaches' can be seen in Table 2.3.

Table 2.3: Comparison of approaches based on the XSD Schema

Approaches	Inputs	Outputs
OWLMAP [83]	XML schema + XML instances	OWL schema + RDF graph
XML2OWL [84]	XML schema + XML instances	OWL schema + individual
XS2OWL [85]	XML schema	OWL schema
XSD2OWL [86]	XML schema + XML instances	OWL ontology + individual
X2OWL [87]	XML schema	OWL schema
Janus [88]	XML schema	OWL schema
EXCO [89]	XML schema + XML instances	OWL schema + individual
Yahia et. al. [90]	XML schema	OWL schema + individual

The correspondence or matching rules are involved throughout the transformation process of XML data to OWL ontologies. They achieve three main objectives: the *generation*, *enrichment,* and *population* of an OWL ontology. The ontology-enriching process from an XML document adds new constructors (classes, object attributes or data types, etc.) to the schema of an existing ontology. In the case of a non-existent ontology, the ontology is generated directly from the XML documents using predefined rules. The process is named the ontology generation process. The ontology population process adds individuals or attributes to available individuals from an XML data to the ontology. Ontology generation and enrichment can be processed using XML instances or validation schemes.

Consequently, two transformation approaches are distinguished to process the generation and the enrichment of ontologies, namely the instances approach and the validation approach. Regarding the population correspondence rules, they mainly require XML document instances. Figure 2.7 shows the different strategies for transforming XML to OWL used by distinct approaches. Two levels are described. The lower level is the instance level, and the upper level is the schema level. On the left, the Figure 2.7 shows the different kinds of XML data, and on the right, it shows the impact on the ontologies; the generation of the ontology, the enriched target ontology and the creation of instances in the target ontology.

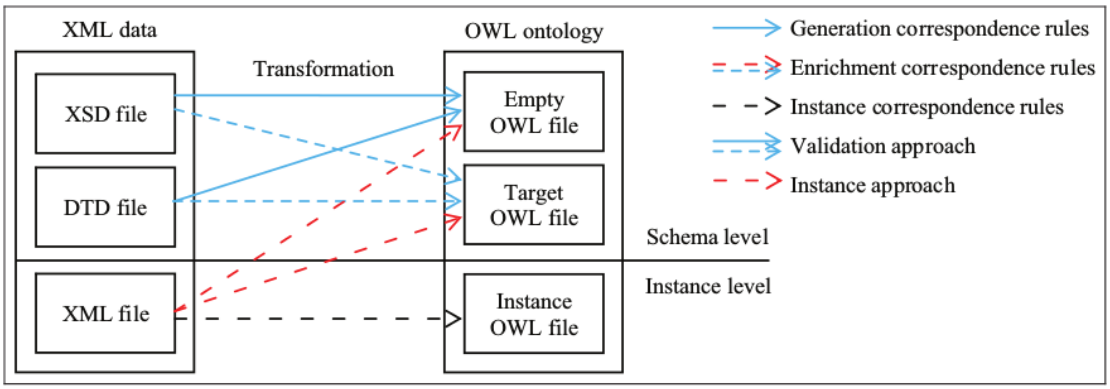


Figure 2.7: Approach and rules for XML to OWL transformation

The arrows symbolize the different processes using correspondence rules, and the two different approaches, the validation, and the instance approach. It should be noted that only XML instances are used for correspondence rules. Based on various mapping approaches, a method called XML2OWL proposed by [84] is used for the basis of data mapping as the focused is currently given to populate new instances to the ontology.

2.6. Semantic Reasoning and Knowledge Retrieval

2.6.1 Rule-based Systems

A rule-based system consists of a database management system for handling the domain-specific facts, a rule set for representing the knowledge structure and relations, and a rule interpreter to carry out the problem solving [91]. Having a knowledge base consisting of facts and rules, a rule interpreter to match the rule conditions against the facts, and a means for extracting the rules, then new knowledge can be derived. Rules are written in Semantic Web Rule Language (SWRL). The rules in SWRL are implication rules and follow this syntax: antecedent → consequent. This form means that the consequent must be true when the antecedent is satisfied. In the SWRL rules, the symbol ∧ means conjunction, ?x is a variable, → means implication. A symbol without the leading '?' denotes the name of an instance (an individual) in the ontology.

2.7. Summary

This chapter has introduced fundamental aspects of this study, which includes ontology-based knowledge representation, foundational ontology for ontology modelling, information extraction approaches for visual and textual analysis, meta data mapping formats and transformation, and rule-based system. All of these aspects are implemented in Chapter 3, where each module of the proposed integrated framework for the security domain surveillance system is extensively presented.

CHAPTER 3

A FRAMEWORK FOR AUTOMATED MEDIA ANALYSIS IN A SECURITY DOMAIN

This chapter highlights the proposed security domain surveillance system framework which incorporates four main modules of the system: (i) data source module; (ii) content extraction module; (iii) parsing module and (iv) semantic knowledge module. The data source and content extraction module focus on data acquisition and salient information extraction through visual analysis of video footage and textual analysis of social media content. The parsing module handles inter-level data transformation and semantic knowledge module perform higher-level event representation, knowledge reasoning, and queries. This framework enables a large amount of video and social media data to be analysed systematically and automatically.

3.1. Related Work

Calavia et. al. [10] proposed an intelligent video surveillance system based on semantic reasoning and ontologies which is able to detect and identify abnormal and alarming situations by analyzing object movement. They implemented a surveillance system based on a three-stage architecture: Sensing, Route Detection and Semantic Reasoning. This architecture forms the foundation of the system framework presented in this thesis. A number of key features are shared by both system frameworks. In sensing stage, the sensor network comprises of smart surveillance cameras that monitors the objects in a region. The frame pre-processing is done to process information extracted from the sensor. Additionally, both systems implemented semantic reasoning, which performs the semantic interpretation of the input data according to the domain knowledge model. This implies that both systems involves the creation of an ontology and a set of semantic rules which describes the domain of knowledge where the system operates.

While both systems centralised in surveillance application, Calavia et. al. are more focused in knowledge domains related to traffic control in a smart city, whereas the main concern in this study is a security domain. Both systems do not perform object identification directly over the video stream. However, Calavia et. al. utilized cameras that run motion detection algorithms to transform the video stream into data packets (XML files) that contain information about the different moving objects. As for this study, our framework implemented object detection algorithm and manual annotation to detect objects and salient features from the video, and executed parsing process to transform the data to XML files. This allows additional features to be extracted based on system requirements and also produced a more robust system. In addition to that, this study integrated textual analysis of social media platform in the system framework, as a supplementary source of event information to complement the knowledge acquired form visual analysis of video footage. This is one of the novel research contribution of this thesis.

3.2. Outline of the proposed framework

Various types of sensors are used to draw information about the situation of an event. For surveillance purpose, CCTV cameras are highly employed for monitoring. A critical aspect of handling CCTV footage is the significant amount of video data that needs to be captured, transmitted, processed and stored. In this context cutting edge technology related to video coding and transmission is a key element of a wholistic semi-automated surveillance system for security. Related work can be found in [92], [93], [94], [95]. There are various types of CCTV surveillance camera today, with many different features and options. The camera may be focused on a fixed location, set to scan a particular area, or they can be operated remotely by specially trained operators. Public-space CCTV camera systems act as aids in public safety deployment decisions and in the identification and subsequent arrest of suspects. CCTV can also work as a deterrent to criminal and socially offensive behaviours, and as evidence gathering tools. CCTV surveillance camera offers a real-time update of events and provides date and time stamped on surveillance footage which is crucial to help keep track of a chain of relevant events.

Ontology-based approach is an effective way to support semantic analysis of multimedia content for event detection and understanding in surveillance domain. Video footages generated by surveillance cameras along with text posts created by social media users contribute to a valuable source of information on real-world events. The combination thus offers a great benefit in the context of surveillance and investigation. During a critical event, both data resources provide real-time reports about on-the-ground situations complemented with time-stamped, geo-located, context-specific information to help security forces understand the extent and severity of events. Therefore, the system aims to exploit the synergy between visual and textual information to achieve broad insight about multimedia content and thus encourage semantic understanding of content.

The proposed ontology-based security domain surveillance system framework consists of *Data Source*, *Content Extraction*, *Parsing* and *Semantic Knowledge* modules as shown in Figure 3.1. It is a system that lies at the crossroads of Visual Analysis and Natural Language Processing (NLP) [96]. Event information is provided by two sources: CCTV surveillance camera as well as the social media network.

Salient information from both sources is extracted and processed in *Content Extraction Module* where objects, actions and other prominent features from video footage and user-generated content from social media text posts are retrieved. The process requires information extraction approaches combining several different techniques, ranging from video analysis to NLP. Techniques and approached used in *Content Extraction Module* is introduced in this chapter and experimental results for both sub-modules (visual and textual analysis) are presented and discussed in Chapter 5 and Chapter 6.

This information is handed to the *Parsing* module which performs data transformation and ontology population process. The parsing process bridge the semantic gap between low-level data and higher level descriptions [97] to support semantic analysis. Implementation of the parsing process is detailed in Chapter 5, Section 5.3.

Consequently, the *Semantic Knowledge Modul*e encodes these semantic data in an ontology using machine-understandable format and reasoning rules were created to support rule-based classification and semantic query of the inferred knowledge. The retrieved information will facilitate the user in interpreting semantic contents from both data sources, and thus assist in multimedia content understanding. Security domain ontology development is presented in Chapter 4 and implementation of rules and semantic reasoners for both visual and social media analysis are demonstrated in Chapter 5 and Chapter 6.

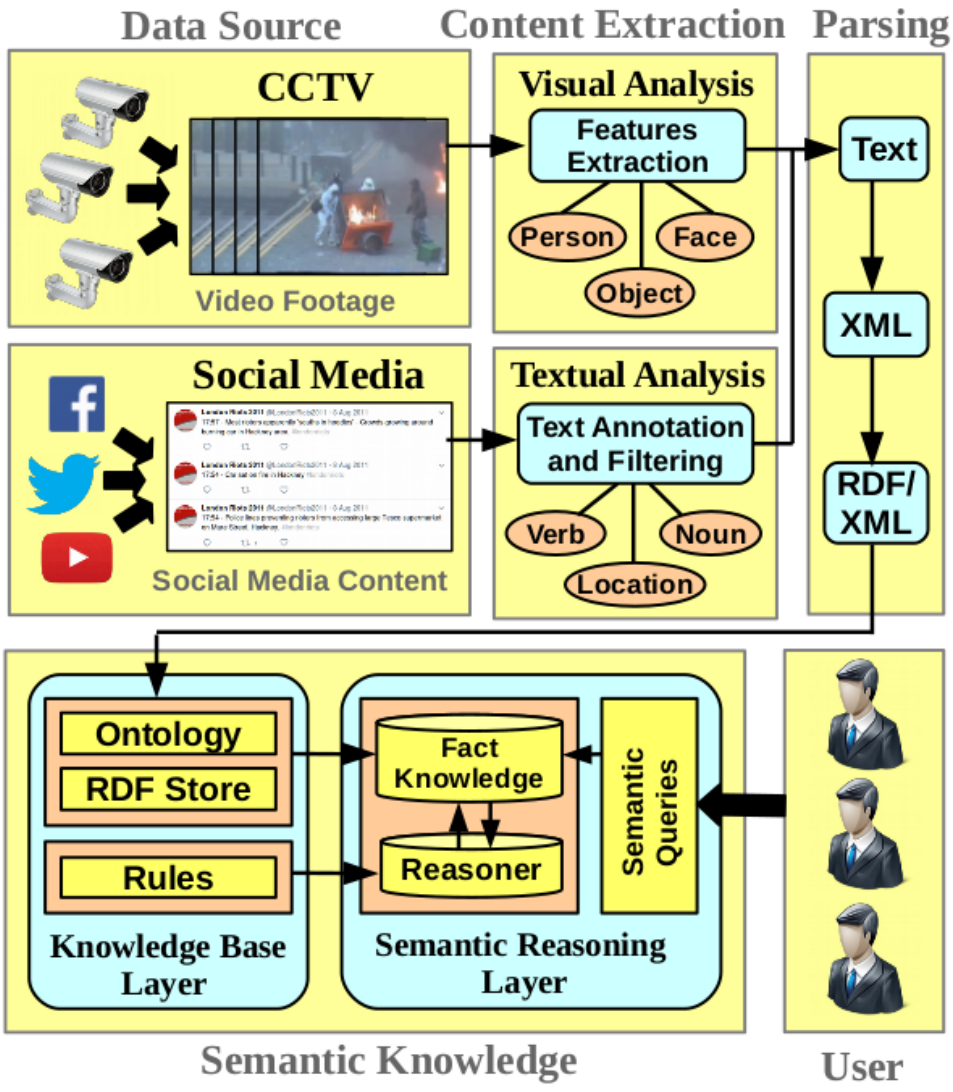


Figure 3.1: Security domain surveillance system framework

3.3. Data Source Module

Various types of sensors are used to draw information about the situation of an event. For surveillance purpose, CCTV cameras are highly employed for monitoring. There are various types of CCTV surveillance cameras today, with many different features and options. The camera may be focused on a fixed location, set to scan a particular area, or they can be operated remotely by specially trained operators. Public-space CCTV camera systems act as aids in public safety deployment decisions and in the identification and subsequent arrest of suspects. CCTV can also work as a deterrent to criminal and socially

offensive behaviours, and as evidence gathering tools. CCTV surveillance camera offers a real-time update of events and provides date and timestamped on surveillance footage which is crucial to help keep track of a chain of relevant events.

On the other hand, the rapid increase of social media usage in recent years has become another important source for reporting real-world events. Effective and efficient event monitoring is made possible through extensive reporting by an active and ubiquitous community [22] of social media users. It is apparent that social media user-generated content present more focused and comprehensive event annotations instead of mere videos or photos. Social media users would produce multi-perspective, multimodal user-generated contents in the form of descriptive text-posts and event-oriented digital photos or videos. These nearly-real-time reports about on-the-ground situations such as locations, times and incidents, have immense value for security forces and emergency authorities to assess events.

Based on these factors, CCTV surveillance camera footage and social media user-generated content are chosen as an essential data source to be analysed in the security domain surveillance system. Dataset for visual semantic analysis is explained in Chapter 5, Section 5.1 and dataset for social semantic analysis is presented in Chapter 6, Section 6.2. By exploiting information from both data sources, ‘the big picture’ during critical situations will be better understood, and thus help emergency authorities make the best decisions possible for deploying aid, rescue and recovery operations during the event [20].

3.4. Content Extraction Module

The function of this module is to extract and process important information from visual and social media sources using various information extraction

approaches where objects, actions and other prominent features from video footage and social media user-generated content are retrieved.

3.4.1 Visual Analysis

Surveillance footage that is acquired from CCTV cameras are processed in the visual analysis module to extract salient features from the video footage. Considering the importance of features in representing an event, feature extraction is an essential step towards higher-level semantic analysis. Prior feature extraction, an important concept of features has been identified and defined in the domain ontology to better represent the event scene in the ontology. For instance, people involved in the event scene are represented by concepts such as *person*, *police* and *crowd*, mobile object is represented by *vehicle* and action is represented by *running* and *kicking*. These identified concepts became the main features that are extracted from the video footages. Detailed descriptions of security domain ontology concepts are presented in Chapter 4.

Several algorithms and approaches are implemented for features extraction based on type of feature to be extracted. In this thesis, Histogram of Oriented Gradient (HOG) feature descriptor [64] is implemented for person detection task and Haar feature-based cascade classifier [63] is adopted for face detection. However, since existing method is not 100% accurate, manual annotation using ViPER-GT annotation tool is implemented to extract feature descriptors of several objects and actions. This includes *vehicle*, *fire* and *throwing* action among others. All feature descriptors obtained in visual analysis module through automated and manual annotations are represented in the domain ontology. These inputs create a knowledge base in the ontology and support the semantic analysis of events in semantic knowledge module [98].

3.4.1.1 Person Detection using Histogram of Oriented Gradient

The HOG feature descriptor as proposed by Dalal and Triggs in [64] is implemented for person detection task. The method is based on evaluating well-normalized local histograms of image gradient orientations in a dense grid. The core idea is that local object appearance and shape can often be characterized by the distribution of local intensity gradients or edge directions, even without precise knowledge of the corresponding gradient or edge positions. This is implemented by dividing the image window into small spatial regions called ‘cells’, for each cell accumulating a local 1-D histogram of gradient directions or edge orientations over the pixels of the cell. The combined histogram entries form the representation. For better invariance to illumination, shadowing, etc., Dalal and Triggs suggested to contrast-normalize the local responses by accumulating a measure of local histogram ‘energy’ over somewhat larger spatial regions block and using the results to normalize all of the cells in the block. The normalized descriptor blocks are referred to as HOG descriptors. The detection window is tiled with a dense grid of HOG descriptors and the combined feature vector is used in a conventional SVM based window classifier to give the human detection chain.

Person detection task is executed on surveillance footage to detect the person involved in the event. Every detected person is marked using a bounding box which is created simultaneously (hard-coded) during the execution of person detection algorithm. Example of person detection can be seen in Figure 3.2. Every bounding box carries information of the number of detected person, the corresponding frame number, four bounding box borders and its centre point coordinate, video capture timestamp (indicates the time, in milliseconds (relative to the starting time) the person being detected after the algorithm has been executed) and video timestamp (date, time and location) as shown below Figure 3.2. These descriptors and image of detected person are used for analysis purpose.

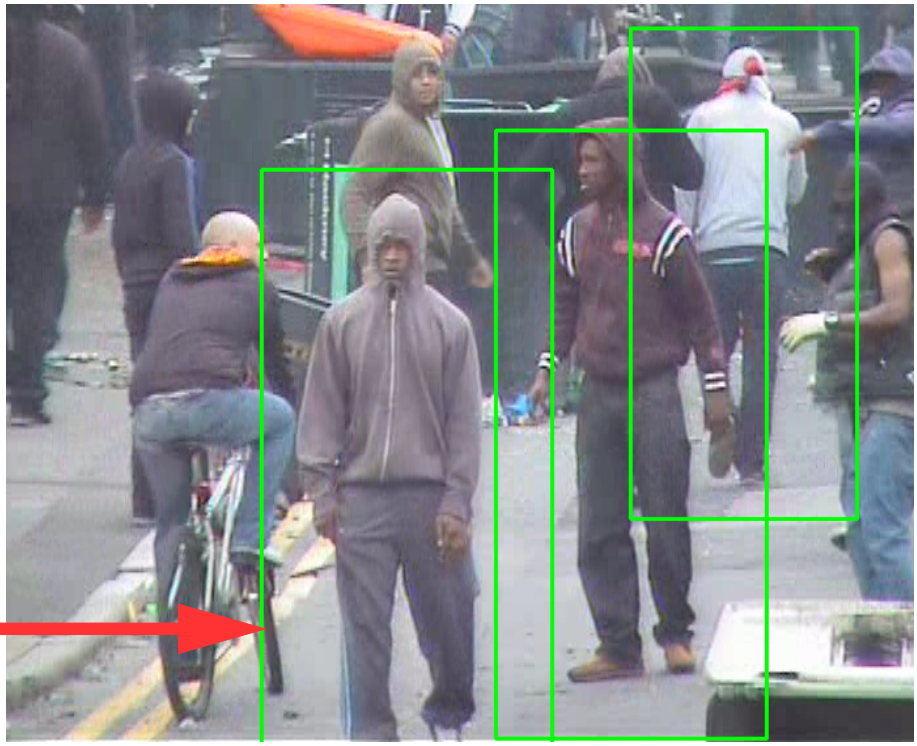


Figure 3.2: Example of bounding boxes created during person detection task

```
Person-2151.png
Frame_number-623 623
Left: 199 Right: 421 Top: 133 Bottom: 604
PersonCenter_X: 310 PersonCenter_Y: 368
Position: 49753.4ms
DateTime: 2011-08-08T18:45:34
Location: Hackney Street: Clarence/Hindry
```

3.4.1.2 Face Detection using Haar feature-based Cascade Classifier

The face detection task performs estimation of face features in each frame using Haar feature-based cascade classifier proposed by Viola and Jones [63], [67]. Their algorithm uses five Haar-like features or kernels (refer Section 2.4.2.1) and calculates all possible sizes and locations of each kernel in the image. This computationally expensive process is solved by introducing the concept of integral image (refer Section 2.4.2.2) to simplify calculation of the

sum of pixels. To select the best features and discard irrelevant ones, the Adaboost approach (refer Section 2.4.2.3) is used. Each feature is applied on all the training images. For each feature, the best threshold which will classify the faces to positive and negative is identified. Features with the minimum error rate, which best classifies the face and non-face images are selected. The final classifier is a weighted sum of these weak classifiers.

However, applying Viola-Jones's 6000 best features on 24x24 window and run through the whole frame would be inefficient and time-consuming. Therefore, they introduced the concept of a cascade of classifiers (refer Section 2.4.2.4). Instead of applying all 6000 features on a window, the features are grouped into different stages of classifiers and are applied one-by-one. The window is discarded if it fails the first stage. If it passes, the second stage of features are applied, and the process continues. The window which passes all stages is identified as a face region.

The face detection task is executed on surveillance footage to detect a person's face in the event. A face is an important feature to recognize the suspect in an event. The face feature is marked using a circle in the video frame. The marking process is done simultaneously (hard-coded) during the execution of face detection algorithm. Example of person detection can be seen in Figure 3.3. Every marked circle carries information of the number of person detected, the corresponding frame number, the centre point coordinates of the circle, video capture timestamp and video timestamp as shown below Figure 3.3. Using this approach, successful detection can be seen on frontal and upright faces. With more rotation (toward a profile view), the detector becomes unreliable. Harsh backlighting in which the faces are very dark while the background is relatively light sometimes causes failures. This approach also fails on significantly occluded faces. The face with covered mouth will usually still be detected. However, occluded eyes usually cause failures.

Figure 3.3: Example of marked face during face detection task

```
Face-527.png
Frame_number-623 623
Center: [292, 197]
Position: 49673.6ms
DateTime: 2011-08-08T18:45:34
Location: Hackney Street: Clarence/Hindry
```

3.4.1.3 Manual Annotation using ViPER-GT

Manual annotations are implemented using Video Performance Evaluation Resource (ViPER) tool for action recognition. ViPER-Ground Truth (ViPER-GT) provides the process of authoring ground truth through frame-by-frame mark up of video metadata. In this research, ViPER-GT is used to complement features extraction approach through manual annotation of objects and actions in the surveillance footage. Annotation executed using ViPER-GT tool produces an XML-based file format to define and instantiates descriptors based on the annotation task. A descriptor is a record describing element of the video which

represents an object that conforms to a user-defined schema. The descriptor composed of descriptor type, descriptor name, attribute type, attribute name and its instances. Every descriptor has a unique ID and an associated span in which it is valid. Example of ViPER-GT annotation is illustrated in Figure 3.4 and excerpt of the corresponding descriptor file is shown in Figure 3.10.

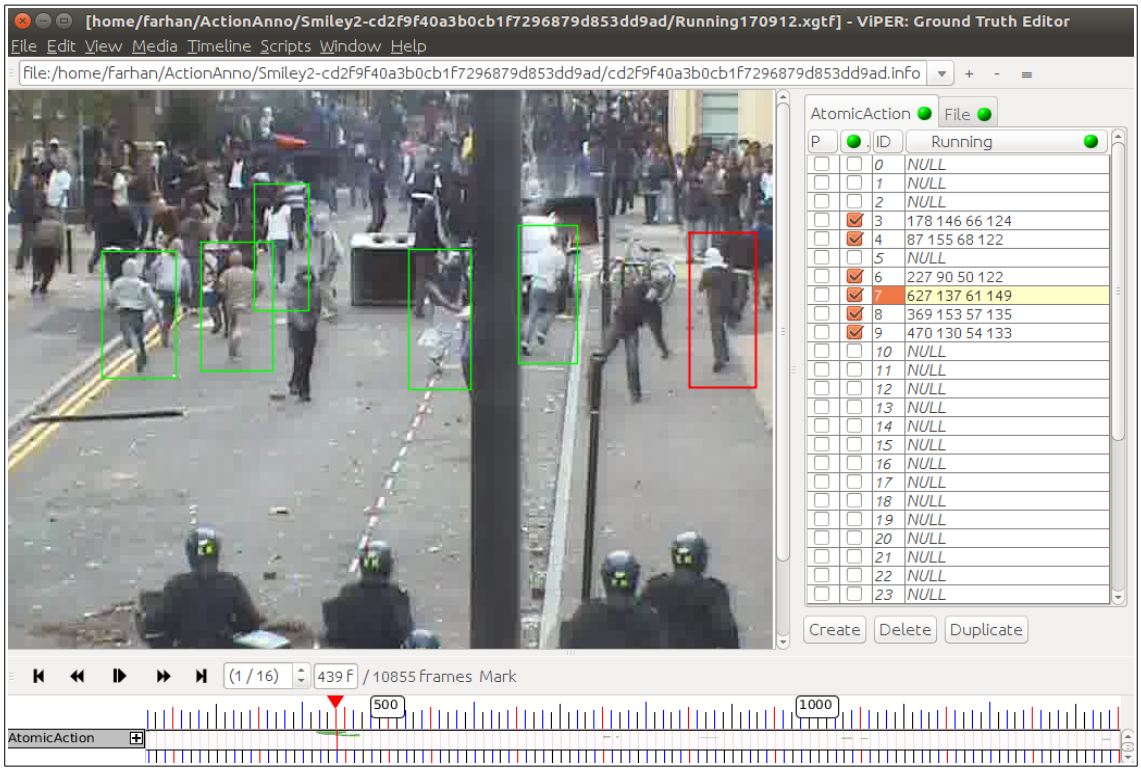


Figure 3.4: Example of ViPER-GT annotation

```
<descriptor name="AtomicAction" type="OBJECT">
  <attribute dynamic="true" name="Running" type="http://lamp.../viperdata#bbox"/>
</descriptor>
<object framespan="421:465" id="7" name="AtomicAction">
  <attribute name="Running">
    <data:bbox framespan="421:425" height="155" width="61" x="635" y="170"/>
    <data:bbox framespan="426:430" height="145" width="61" x="635" y="170"/>
    <data:bbox framespan="431:432" height="145" width="61" x="632" y="159"/>
    <data:bbox framespan="433:435" height="145" width="61" x="634" y="152"/>
    <data:bbox framespan="436:440" height="145" width="61" x="627" y="137"/>
    <data:bbox framespan="441:441" height="136" width="61" x="627" y="137"/>
    …
    …
    <data:bbox framespan="463:465" height="128" width="61" x="620" y="190"/>
  </attribute>
</object>
```

Figure 3.5: Excerpt of descriptor file from ViPER-GT annotation

In this example, descriptor name is given as '*AtomicAction',* descriptor type is '*Object',* attribute name is '*Running'* and attribute type is set to be '*bbox'* which indicate the bounding box (Figure 3.5). '*Object'* descriptors refer to an object that may have many instances at any given time, and whose instances may change over time. Each bounding box represents an instance and carries information about its location in the video frame. For example, the red bounding box in Figure 3.4 refers to a location of an object with ID='7' in the video frame. The following four numbers on the right represent the height, width and top-left coordinate (x,y) of the bounding box. There were six bounding boxes in Figure 3.4 indicated by six different IDs in the table on the right.

3.4.2 Textual Analysis

3.4.2.1 Text Annotation

GATE is an infrastructure for developing and deploying software components that process human language. GATE is distributed with an Information Extraction system called A Nearly-New Information Extraction (ANNIE) system [70]. Using the ANNIE plugin, the annotation process in GATE follows through a corpus pipeline of ANNIE resources. The annotation process using GATE software is illustrated in Figure 3.6.

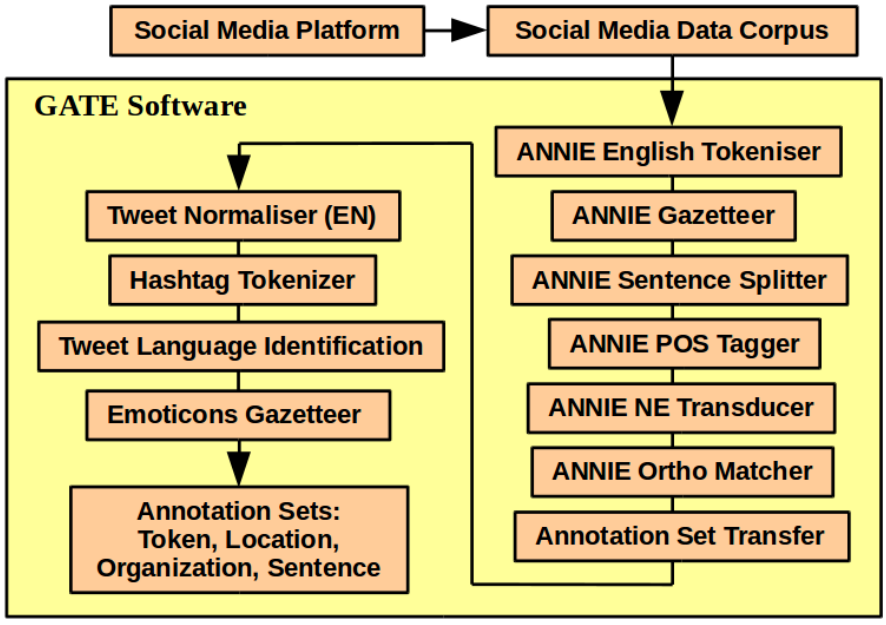


Figure 3.6: Textual data annotation pipeline using GATE software

To annotate Twitter data corpus, a Twitter plugin which contains additional resources needed for Twitter data analysis is included. Each article in the corpus is linguistically pre-processed by performing fine-grained tokenization, gazetteer, sentence splitting, Part-of-Speech (POS) tagging, Named Entities (NE) transducing, Ortho matching, Tweet normalising, Hashtag tokenization, Language Identification, and Emoticons gazetteer to produced annotation sets.

Tokeniser

The tokeniser [37] splits the text into very simple tokens such as numbers, punctuation, and words of different types. Token types can be classified into word, number, symbol, punctuation, and space token. A word is defined as any set of contiguous upper or lowercase letters, including a hyphen and a number is defined as any combination of consecutive digits. Symbol and punctuation are divided into several types. Two types of symbol are defined as currency symbol (e.g. '$', '£') and symbol (e.g. '&', '^'), whereas three types of punctuation are defined as start punctuation (e.g. '('), end punctuation (e.g. ')'), and other punctuation (e.g. ':'). Space token is another type of token which is determined by white spaces in the corpus. The English Tokeniser is a processing resource that comprises a normal tokeniser and a Java Annotation Patterns Engine (JAPE) transducer. The transducer has the role of adapting the generic output of the tokeniser to the requirements of the English POS tagger.

Gazetteer

The role of the gazetteer [99] is to identify entity names in the text based on lists. The gazetteer lists used are plain text files, with one entry per line. Each list represents a set of names, such as names of cities, organisations, days of the week, etc.

Sentence Splitter

The sentence splitter is a cascade of finite-state transducers which segments the text into sentences. This module is required for the tagger. The splitter uses a gazetteer list of abbreviations to help distinguish sentence-marking full stops from other kinds. Each sentence is annotated with the type ‘Sentence’.

Part of Speech (POS) Tagger

The tagger is a modified version of the Brill tagger, which produces a part-of-speech tag [37] as an annotation on each word or symbol. The tagger uses a default lexicon and ruleset. Two additional lexicons exist - one for texts in all uppercase (lexicon cap), and one for texts in all lowercase (lexicon lower). The default lexicon should be replaced with the appropriate lexicon at load time to use these. The default ruleset should still be used in this case.

Orthographic Coreference (OrthoMatcher)

The Orthomatcher module adds identity relations between named entities [100] [99] found by the semantic tagger, in order to perform coreference. It does not find new named entities as such, but it may assign a type to an unclassified proper name, using the type of a matching name. The matching rules are only invoked if the names being compared are both of the same type, i.e. both already tagged as (say) organisations, or if one of them is classified as ‘unknown’. This prevents a previously classified name from being recategorized.

Annotation Set Transfer

The Annotation Set Transfer [70] allows copying or moving annotations to a new annotation set if they lie between the beginning and the end of an annotation of a particular type. For example, this can be used when a user only wants to run a processing resource over a specific part of a document, such as the Body of

an HTML document. The user specifies the name of the annotation set and the annotation which covers the part of the document they wish to transfer, and the name of the new annotation set. All the other annotations corresponding to the matched text will be transferred to the new annotation set.

3.5. Parsing Module

Parsing module aims to transform syntactic information obtained from the content extraction module to high-level semantic concept representation in the semantic knowledge module. The parsing module performs a transformation process from text instance document to RDF/XML ontology and populates existing OWL ontology with newly generated instances. The mapping is implemented in the standard XML technology, XSLT which raises the XML source documents to the level of an OWL ontology.

3.5.1 XML and OWL Model

XML to OWL transformation process interprets the tree structure of XML and represents the intended model in the OWL model which is based on the subject-predicate-object structure from RDF/RDFS [71]. In order to apply a semantic meta-information for reasoning on instance data, XML documents have to be mapped to RDF, bridging the gap between those models. The first part of the concept concerns the XML to RDF mapping.

XML is a language that defines a generic syntax to store and exchange documents by means of a tree-based structure. Although RDF has an XML-based syntax, XML and RDF serve different purposes and have been developed separately within the W3C, which lead to different modelling foundations. XML is based on a tree model where only nodes are labelled, and the outgoing edges are ordered. This model originates from semi-structured data and databases. In contrast to this, RDF is based on a directed graph

model where edges have labels but are unordered. Figure 3.7 and 3.8 illustrate the difference between XML and RDF where RDF distinguishes between resources (e.g. *Face-1*) and properties (e.g. *hasFrameNumber*, *hasFaceCenter_X*) while XML does not (e.g. both would be elements). Details regarding textual file generation, text to XML data conversion and XML to RDF mapping is presented in detail in Section 5.3.1 to 5.3.3.

```
<Face>
    <NamedIndividual>Face-1</NamedIndividual>
    <hasFrameNumber>Frame_number-2</hasFrameNumber>
    <hasFaceCenter_X>666</hasFaceCenter_X>
    <hasFaceCenter_Y>35</hasFaceCenter_Y>
    <hasDateTime>2011-08-08T18:45:34</hasDateTime>
    <hasLocationName>Hackney</hasLocationName>
    ...
</Face>
```

Figure 3.7: Fragment of an XML document instance

```
<owl:NamedIndividual xmlns:owl="http://www.w3.org/2002/07/owl#"
  rdf:about="http://www.semanticweb.org/farhan/ontologies/ForensicDomain#Face-1">
  <rdf:type rdf:resource="http://www.semanticweb.org/farhan/ontologies/ForensicDomain#Face"/>
  <ForensicDomain:hasFrameNumber xmlns:ForensicDomain="http://www.semanticweb.org/farhan/ontologies/
      ForensicDomain#"
      rdf:resource="http://www.semanticweb.org/farhan/ontologies/ForensicDomain#Frame_number-2"/>
  <ForensicDomain:hasFaceCenter_X xmlns:ForensicDomain="http://www.semanticweb.org/farhan/ontologies/
      ForensicDomain#"
      rdf:datatype="http://www.w3.org/2001/XMLSchema#int"
       xml:base="http://www.semanticweb.org/farhan/ontologies/ForensicDomain#">666
       </ForensicDomain:hasFaceCenter_X>
  <ForensicDomain:hasFaceCenter_Y xmlns:ForensicDomain="http://www.semanticweb.org/farhan/ontologies/
      ForensicDomain#"
      rdf:datatype="http://www.w3.org/2001/XMLSchema#int"
       xml:base="http://www.semanticweb.org/farhan/ontologies/ForensicDomain#">35
       </ForensicDomain:hasFaceCenter_Y>
   …
   …
</owl:NamedIndividual>
```

Figure 3.8: Types of element indicated by labels in RDF documents

3.5.2 Parsing Framework

To bridge the gap between different data representation, a procedure that transforms XML documents to OWL ontology is developed in this study. Figure 3.9 shows the sequence in the parsing process. First, a text file consisting of a list of newly extracted features and text instances generated from content

extraction module (Section 3.4) is parsed into a hierarchical XML instance document using Simple API for XML (SAX) parser. Next, an RDF/XML instance is generated from an XML instance document using XSLT in an XSLT processor. A set of template rules are implemented in XSLT stylesheet to transform the source XML into RDF/XML document (see Figure 5.12). XSLT stylesheet that has been produced is used to automatically generate the desired ontology. In the final step, the RDF/XML instance document is merged with the OWL model that has been created using Protégé tools using the Jena framework. Jena is a Java API which supports the creation and manipulation of RDF graphs to represent resources, properties, and literals in RDF/XML and OWL [101]. The updated OWL ontology contains information of new instances obtained from the the content extraction module. To support the separation of model and data, the OWL model is created separately from the RDF/XML instances. The framework is designed to be easily extensible so that better support for document-oriented XML can be integrated. Detail sequence of the parsing process is elaborated in Chapter 5, Section 5.3.

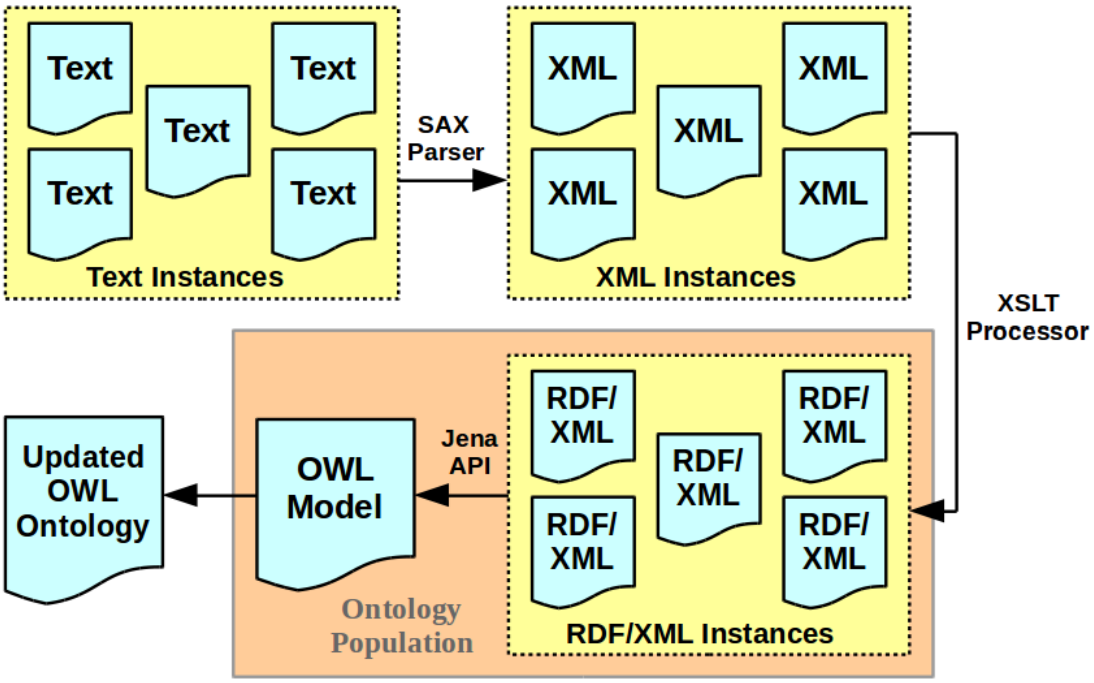


Figure 3.9: Sequence in the parsing process

3.6. Semantic Knowledge Module

The Semantic Knowledge module manages all the semantic operations. It consists of a knowledge base layer and semantic reasoning layer. In the knowledge base layer, a domain ontology and RDF store [102] are employed for high-level representations, and rules are created for rule-based classifications. The semantic reasoning layer consists of a fact knowledge and reasoner for knowledge inferences. Semantic queries are responsible for knowledge queries, which are used to retrieve the in-memory triples [103] in the newly inferred knowledge.

3.6.1 Ontology

The purpose of the ontology is to formalize the basic concepts, attributes of concepts and the relationships between concepts in the domain of discourse. Security domain ontology has been built to represent an events model in a broad variety of forensic context.

3.6.2 Rules

A rule-based system is used to store and manipulate knowledge to interpret information in a useful way. The set of semantic rules are a formal specification of conditions and logic operations to be performed over the ontology to draw conclusions from the data. Semantic rules are specified using SWRL [104] [105] which allows users to write rules that can be expressed in terms of OWL concepts to provide more powerful deductive reasoning capabilities. SWRL is a standard language based on OWL-DL and on the Rule Markup Language (RuleML) which provides both OWL-DL expressivity and rules from RuleML [106]. SWRL rules are adopted to build reasoning rules in order to represent the dynamic aspect of the surveillance system. During reasoning, inferences are made, classifying the instances of the security domain ontology and associating new properties to instances while maintaining logical consistency [107].

3.6.3 Reasoner

A semantic reasoner, reasoning engine, rules engine, or simply a reasoner, is a piece of software able to infer logical consequences from a set of asserted facts or axioms. Many applications developed for the semantic purpose requires some kind of reasoning capability. This is because the intermediate metadata comes with uncertainty [108] which affects the acceptable accuracy and robustness for a semantic complex query. Providing sound and complete reasoning services are essential for many of these applications to function properly. Pellet [109] is the first sound and complete OWL-DL reasoner with extensive support for reasoning with individuals (including nominal support and conjunctive query), user-defined datatypes, and debugging support for ontologies.

Pellet, in its core, is a DL reasoner. However, unlike other DL reasoners, it has been designed to work with OWL right from the beginning. This design choice had a huge influence on the overall architecture. It affected how the tableaux reasoner was implemented, e.g. with the ability to reason with instance data (ABox reasoning) without making the Unique Name Assumption, and what kind of supporting modules to have, e.g. having an XML Schema datatype reasoner and a query engine.

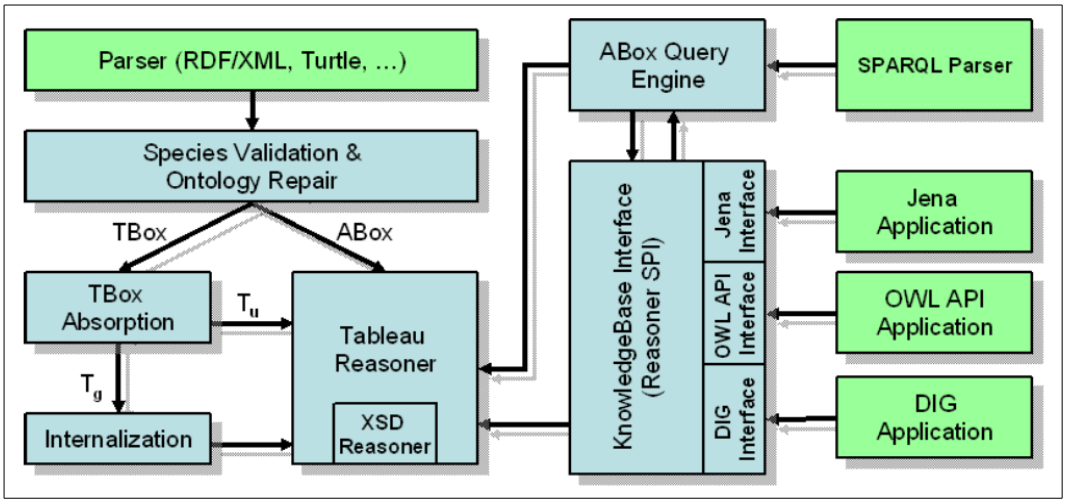


Figure 3.10: Main components of the Pellet reasoner

Figure 3.10 shows the main components of Pellet. The core of the system is the tableaux reasoner [103] that checks the consistency of a knowledge base. The reasoner is coupled with a datatype oracle that can check the consistency of conjunctions of (built-in or derived) XML Schema simple datatypes. The OWL ontologies are loaded into the reasoner after species validation and ontology repair. This step ensures that all the resources have an appropriate type triple and missing type declarations are added according to some heuristics. The heuristics implemented in Pellet attempt to guess the correct type for an un-typed resource. These are mainly standard operations, e.g. a resource used in the predicate position is inferred to be a property.

During the loading phase, axioms about classes are put into the TBox component and assertions about individuals are stored in the ABox component. TBox axioms go through the standard preprocessing of DL reasoners, e.g. normalization, absorption and internalization, before they are fed to the tableaux reasoner. The system provides a thin layer for programmatic access through the Service Programming Interface (SPI) that provides convenience functions to access the reasoning services provided.

A practical OWL reasoner provides at least the standard set of DL inference services, such as consistency checking [111] [112]. An ontology is consistent if there is an interpretation that satisfies all the facts and axioms in the ontology. In this project, Pellet reasoner is used to check the ontology design consistency. Tableaux reasoner is the core of the system that checks the consistency of a knowledge base.

3.6.4 Reasoning

Reasoning in ontologies and knowledge bases is one of the reasons why a specification needs to be a formal one. Reasoning with ontologies is an

automatic procedure that infers new axioms which have not been explicitly included in the knowledge base but are logical consequences of the represented axioms [113]. All of the formalisms mostly were created with the outlook of automatic processing, but due to their properties such as decidability or computational complexity or even due to the level of formality, it is not always possible. A few examples of tasks required from reasoner are as follows.

- **Satisfiability** of a concept - determine whether a description of the concept is not contradictory, i.e., whether an individual can exist that would be an instance of the concept.
- **Subsumption** of concepts - determine whether concept *C* subsumes concept *D*, i.e., whether the description of *C* is more general than the description of *D*.
- **Consistency** of ABox with respect to TBox - determine whether individuals in ABox do not violate descriptions and axioms described by Tbox.
- **Check an individual** - check whether the individual is an instance of a concept
- **Retrieval of individuals** - find all individuals that are instances of a concept
- **Realization of an individual** - find all concepts which the individual belongs to, especially the most specific ones

3.6.5 Semantic Queries

SPARQL Protocol and RDF Query Language (SPARQL) [72] [114] is the standard query language and protocol for Linked Open Data on the web or for RDF triplestores. SPARQL enables users to query information from databases or any data source that can be mapped to RDF. The queries are used to retrieve the in-memory triples in the newly inferred knowledge. The SPARQL standard is designed and endorsed by the W3C and helps users and

developers focus on what they would like to know instead of how a database is organized.

3.7. Summary

This chapter introduces and elaborates every module functionality in security domain surveillance system framework. The module consists of Data Source, Content Extraction, Parsing and Semantic Knowledge Module. The contribution lies in the data retrieval and processing in Content Extraction and Parsing module, and ontology modelling and semantic rules formation in the Semantic Knowledge Module. In the next chapter, the development of an event conceptual model for security domain ontology, which is one of the main components in Semantic Knowledge Module is demonstrated. Chapter 5 and 6 follows with experimental analysis of the visual and social media sources for system validation.

CHAPTER 4

SECURITY DOMAIN ONTOLOGY

This chapter presents the development of an event conceptual model for security domain ontology through an implementation of ontology development methodology. It also presents the conceptualization classification extension in accordance with six elementary aspects which underpins functional requirements of an event model. Foundational ontology is used to guide the development of domain ontology to facilitate semantic interoperation between heterogeneous systems.

4.1. Semantic Knowledge Module

The semantic knowledge module manages all the semantic operations in the security domain surveillance system. It consists of domain ontology and RDF store for high-level representations, rules and reasoner for rule-based classifications and inferences, and query engine for semantic queries. Domain ontology has been introduced in Section 2.2.3 and ontology engineering methodology proposed by various researchers has been presented and discussed in Section 2.3.1. Different types of foundational ontologies and its applications have also been summarised in Section 2.3.2.

Ontology is regarded as a fundamental component that fabricates the Semantic Knowledge Module. As shown in Figure 4.1, ontology is one of the main components that make up the Knowledge Base Layer. In order to construct an ontology, five phases are implemented in ontology development methodology. The subsequent sections will focus on the details.

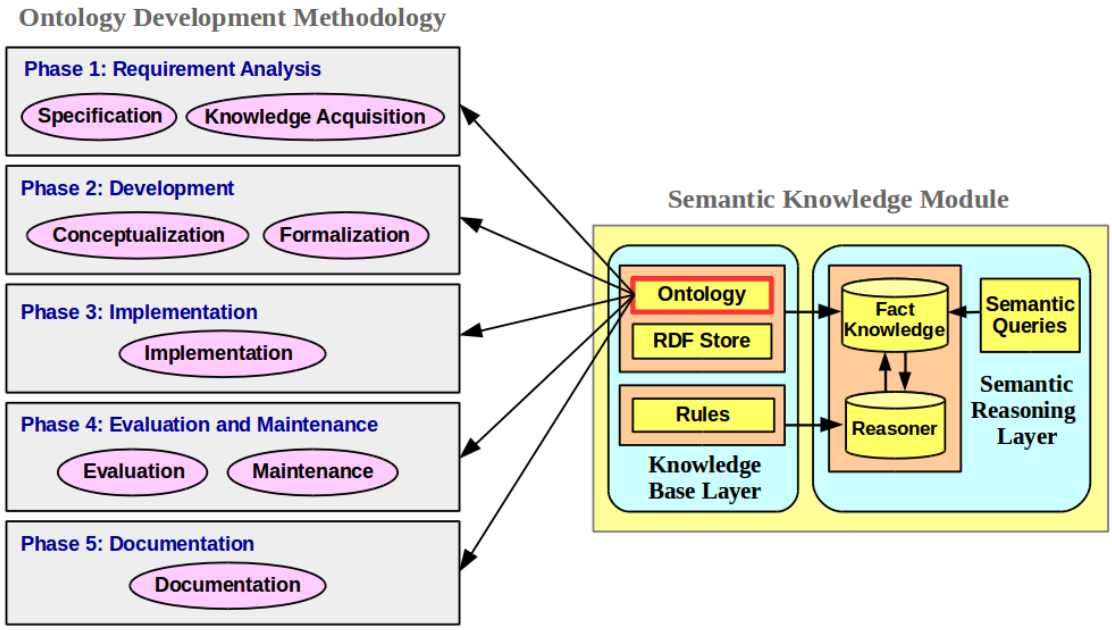


Figure 4.1: Ontology development methodology phases

4.1.1 Ontology Development Methodology

One of the critical issues for implementing any ontology is the problem of choosing the mature and the right methodology. Adapting a mature methodology will enhance the quality of the implemented ontology. Based on the discussion on ontology development methodology presented in [48]-[55], several phases are adapted to serve as a guideline for developing security domain ontology as follows:

Phase 1: Requirement Analysis

Specification

This phase involves identifying all specifications of ontology requirements [56] which includes the purpose, scope, target users, use case scenarios, user requirements and ontology requirements in terms of equipment and software. Identifying these criteria is very important as it leads to focusing on the only necessary data to be analysed.

Knowledge Acquisition

The knowledge acquisition phase starts with the procurement process [55]. The activities carried out in this phase are literature survey and analysis of related documents. In this thesis, video content of recorded footage and social media posts of related events are examined and analysed. The concepts represented in the domain ontology are carefully drafted to ensure a broad and useful functionality of the ontology.

Phase 2: Development

Conceptualization

In this sub phase, the conceptual model for security domain is developed. The objective of this activity is to organize and structure the knowledge acquired during knowledge acquisition using external representations that are

independent of the knowledge representation and implementation paradigms in which the ontology will be formalised and implemented next. The model is formulated based on elementary aspects of event description as proposed in [115] which promotes a model that supports a common foundation for a wide diversity of applications, reusability and application integration. Figure 4.2 shows an illustration of basic aspects which incorporates temporal, spatial, informational, experiential, structural and causal aspect of event description.

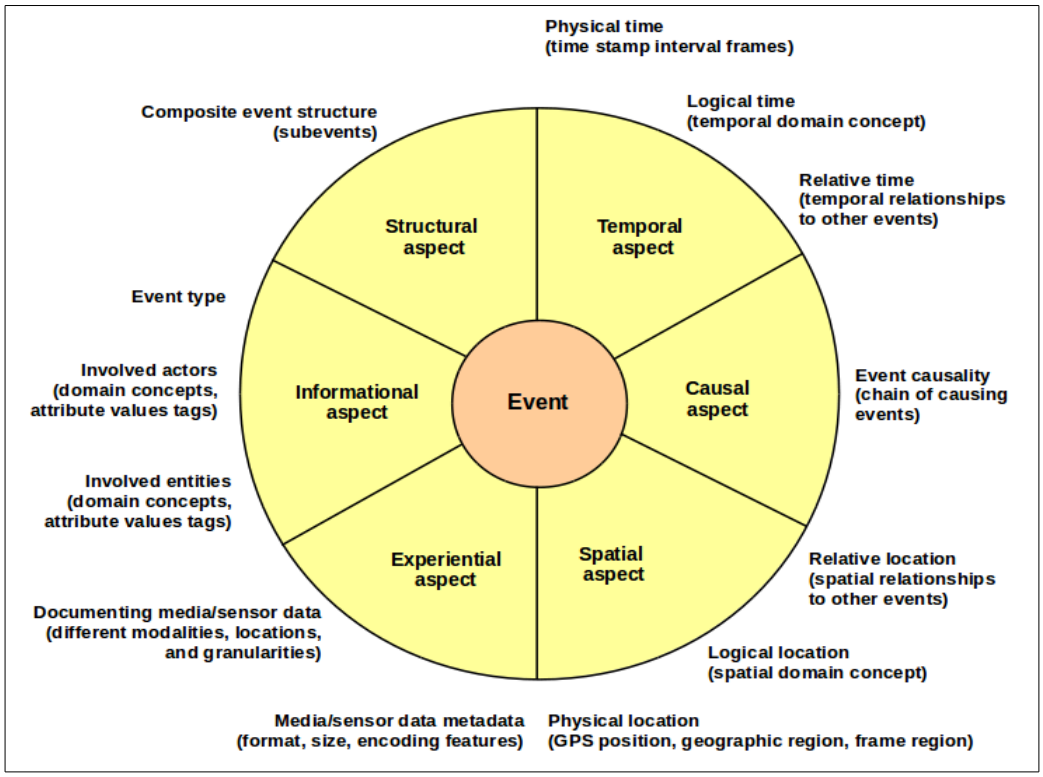


Figure 4.2: Elementary aspects of event description

i. Temporal aspect

The comprehensive temporal aspect of an event incorporates the physical and logical level, and in absolute or relative manner. The time of an event's occurrence could be expressed in a global time measure (for example date, time and time zone), using a relative time measure (such as the game minutes in sporting events), or in relation to media (for example the frame numbers of a video documenting an event).

ii. Spatial aspect

The spatial aspect of an event model shows location awareness and support different ways of capturing a broad level of space. Global, local, and media-related units of measurement are suitable for expressing the spatial aspect on a physical level. On a logical level, applications can express an event’s location not only in an absolute manner but also relative to other event’s locations.

iii. Informational aspect

An event model provides information about the events that occur. Adequate coverage of the event's informational aspect might require further description. This can include the actors and entities involved in an event and their roles. It might also involve further parameters describing the event or the entities. Depending on the application, different methods might be adequate to capture actors and entities involved in an event.

iv. Experiential aspect

An event model must also show media awareness and let events refer to such media. An event model's media referencing scheme should be capable of addressing media of different types, ranging from traditional discrete and continuous media such as images or videos and complex media such as multimedia presentations to essentially any kind of sensor data that is available on the course of events.

v. Structural aspect

Events are a modelling concept that is applicable at many different levels of abstraction. Thus, the exploration of the subevents that occurred as part of a more complex event offers important insights into an event's course. To address these, three kinds of structural relationships between events are being used:

- Mereological relationship represent events that are usually made up of other events [116]

- Correlation refers to two events that have a common cause
- Causal relationship model causes and effects of events and should support the integration and use of different causal theories as discussed, e.g., in [117]

vi. Causal aspect

A suitable common multimedia event model should offer the means to express causality and permit the explicit representation of chains of causal events for individual events. Offering answers about an event's cause is another essential task of many applications.

Formalization

In this phase, the conceptual model was transformed into a formal representation using Protégé (an ontology editing package). During the integration stage, any existing domain ontology is identified by processing parts of the ontology which are appropriate or otherwise. If such ontology was suitable, it would be integrated into the developed ontology.

Phase 3: Implementation

The main aim of this phase is to change the human readable representation to machine-readable representation. RDF is a standard model for data interchange on the Web which is described using Research Description Framework Schema (RDFS) and OWL modelling languages. RDFS allows users to express the relationships among data by standardizing them using a flexible, triple-based format and then providing relevant vocabulary or keywords, such as *“rdf:type”* or *“rdfs:subClassOf”*, which can be used to express such data. On the other hand, OWL is more powerful as it describes data models more efficiently using appropriate database queries and supported by many available reasoners.

Phase 4: Evaluation and Maintenance

The fourth phase involved evaluating and assessing the developed ontology to determine whether it meets the required specifications which is to support retrieval. This is supported by a discussion in [118] which state that the organization of elements in knowledge representation must facilitate the retrieval of useful information. This implementation involved the development of a system prototype using the developed knowledge representation. This phase is challenging as information systems are not easy to be assessed and there are many aspects to be considered in the assessment process.

Phase 5: Documentation

The final phase is ontology documentation. Effective knowledge sharing requires adequate documentation. This phase is very important because ontology can be reused only if it is properly documented. Documentation should be done with utmost care and must record all the assumptions that are made explicitly.

4.1.2 Requirement Analysis

In Section 4.1.1, the ontology engineering process begins with requirement analysis where identifying all specifications of ontology requirements and procurement process related to information of the selected domain is being carried out. The specifications of the ontology presented in this thesis are identified as:

- **Purpose of the ontology**: To represent events for the purpose of investigation in surveillance applications.
- **Scope of the ontology:** Security domain ontology.

- **Target users of the ontology**: Law enforcement agents, security forces, investigators or analysts.
- **Ontology use case scenario**: Criminal offences conducted in the context of riots and the situation during the riot events.
- **User requirements**: To support information retrieval based on keywords that are inserted by users to the semantic application.
- **Ontology requirements of equipment and software**: Protégé 5.1 is used to support ontology development and for formalizing the developed ontology.

Use Case Scenario: Riots

As a working use case, the scenario related to criminal offences conducted in the context of riots and situation during the riot events is explicitly being identified and referred. The scenario description and summary of events can be described as follows:

Narrative: *The riots occurred between 6 and 11 August 2011 in several London boroughs and in cities and towns across England. The resulting chaos generated looting, arson, and mass deployment of police and resulted in the deaths of people. The riot resulted in several violent clashes with police, along with the destruction of police vehicles, a double-decker bus and many homes and businesses, thus rapidly gaining attention from the media. Protests started in Tottenham, London and overnight, looting and rioting took place in other parts of London. With access to Twitter as a communication medium, social media was used to rapidly spread messages of riots. The online video website YouTube soon host video footage of the riots, which has been recorded by witnesses and participants.*

4.1.3 Development

Once the specification and knowledge acquisition process is completed, a security domain ontology concept is identified based on the requirements of conceptual model development. Six elementary aspects of event description including temporal, spatial, informational, experiential, structural and causal proposed in [115], in conjunction with a functional requirements of an event model discussed in [119] are being synthesized into the model to produce comprehensive concepts of the domain. For each requirement, the use case scenario is also explicitly referred. For modelling security domain ontology, careful alignment with the foundational ontology has also been executed to guide the development of the ontology. One primary reason for this is that, by building a domain ontology as an extension of a foundational ontology, all of the relevant semantic content of the foundational ontology can be inherited with minimal effort [69].

4.1.3.1 DOLCE Foundational Ontology

Descriptive Ontology for Linguistic and Cognitive Engineering (DOLCE) [120] [121] is a foundational ontology that is implemented to classify every concept behind the ontological modelling decisions. Foundational ontologies act as a reference that commits to certain theories and provides a set of formal guidelines for domain modelling and serve as a tool for making heterogeneous ontologies interoperate or merge [120]. This alignment could lead to conceptually more rigorous, cognitively transparent, and efficiently exploitable in several applications. DOLCE's most basic categories of particulars are depicted in Figure 4.3.

The OntoClean methodology is used for a formal evaluation of taxonomical structures and is based on philosophical notions [122]. The core of the methodology are the four fundamental ontological notions of *rigidity*, *unity*, *identity*, and *dependence*. By attaching them as meta-relations to concepts in a

taxonomy, they are used to represent the behaviour of the concepts. OntoClean evaluation of DOLCE ontology has classified its basic categories of particulars as *rigid* properties. Therefore, the importance of focusing on these properties first is emphasized [111] and materialized during our ontology modelling. However, OntoClean methodology will not be covered in this thesis and thus, security domain ontology evaluation using OntoClean will be a potential future work.

Figure 4.3: DOLCE's top categories

Classically, the difference between enduring and perduring entities (which are also called endurants and perdurants) is related to their behavior in time. Endurants are always wholly present at any time they are present while perdurants just extend in time by accumulating different temporal parts. Simply put, all endurants proper parts are present at any time they are present whereas, for perdurants, some of their proper parts are only partially present, in

the sense that (for example, their previous or future phases) might not be present [121]. In DOLCE, the main relation between endurants and perdurants is that of participation: An endurant “lives” in time by participating in some perdurant(s). This can be exemplified by a person (which is endurant) who participate in a discussion (which is classified as perdurant).

Enduring entities (*Endurant*) comprise of physical and non-physical endurant, according to whether they have spatial qualities. Within *PhysicalEndurant,* the concepts are distinguished between *PhysicalObject* and *Feature*. Physical objects are endurants with unity. However, they have no common unity because different subtypes of objects can have different unity criteria. Features are essential wholes which generically are constantly dependent on physical objects (their hosts). Some features may be *RelevantParts* of their host, such as a window of a building, or *Places* such as a hole in a piece of cheese, the underneath of a table, the front of a house, which are not parts of their host.

Special recognition is given to intentions, beliefs, and desires within physical objects. These are called *AgentivePhysicalObject* as opposed to *NonAgentivePhysicalObject*. Intentionality is understood here as the capability of heading for, or dealing with, objects or states of the world. Example of non-agentive physical objects are houses, body organs, pieces of wood, and so on. *NonPhysicalEndurant is* divided into *SocialObjects* and *MentalObjects*, according to whether they are ‘produced’ by a single agent or generically dependent on a community of agents [120]. Social objects are further divided into *AgentiveSocialObject* and *NonAgentiveSocialObject*. Examples of agentive social objects are social agents such as “the president of the United States.” Social agents are not constituted by agentive physical objects, but they can constitute societies. Examples of non-agentive social objects are laws, norms, shares, and peace treaties, which are generically dependent on societies.

Perdurants comprise what are variously called events, processes, phenomena, activities, and states which describe entities that happen in time. They can have temporal parts or spatial parts. For example, the first movement of an execution of a symphony is a temporal part of it. However, the play performed by the left side of the orchestra is a spatial part. An occurrence type is *Stative* or *Eventive* according to whether it holds the mereological sum of two of its instances; that is, if it is cumulative or not. A sitting occurrence is stative because the sum of two sittings is still a sitting occurrence. Within stative occurrences, we distinguish between *States* and *Processes* according to homeomericity: Sitting is classified as a state but running is classified as a process because there are (very short) temporal parts of a run that are not themselves runs. Finally, eventive occurrences are called *Achievements* if they are atomic and *Accomplishments* otherwise.

Quality is the basic entities that can be perceived and measured. Qualities are inherent to entities, where every entity comes with certain qualities which exist exactly as long as the entity exist. *PhysicalQualities* are those that directly inhere to physical endurants and *TemporalQualities* are those that directly inhere to perdurants. Finally, the only class of *Abstract* entities in DOLCE is QualityRegions. The main characteristic of abstract entities is that they do not have spatial nor temporal qualities, and they are not qualities themselves. Quality spaces are special kinds of quality regions, being mereological sums of all the regions related to a certain quality type.

4.1.3.2 Security Domain Ontology Conceptual Model

Based upon conceptualization discussion presented in Section 4.1.1, the security domain ontology conceptual model is tailored in conformity with elementary aspects of event description, along with an adaptation of DOLCE foundational ontology in the process of domain modelling.

Informational aspect

In security domain ontology, information about events is mainly represented by **participation of objects** in the event. According to DOLCE foundational ontology, object conceptual model is represented in *PhysicalObject* of *Endurant*. *PhysicalObject* is characterised by two main concepts; *AgentivePhysicalObject* and *NonAgentivePhysicalObject.* Physical objects that have intentionality [121] are called agentive, and those which do not are called non-agentive. Within *AgentivePhysicalObject*, the concept *Human* and *Animal* are defined. The concept *Person*, *Police* and *Crowd* are distinguished as a subclass of human while *Horse* and *Dog* are a subclass of animal. *Person* refers to a single individual, characterized by face and body features, whereas *Crowd* represents multiple people in a group. *NonAgentivePhysicalObject* consist of five concepts which includes *FixedObject, MobileObject, PortableObject, BodyPart and MediaType.* Fixed object represents an object permanently located, build or installed such as *Building, Shop* and *Hospital* while portable object is mainly small objects such as *Bin* and *Bottle.* Mobile object is any movable object largely *Vehicles.* Figure 4.4 shows class hierarchies representing the concept of physical object and Table 4.1 elaborate the informational aspect representation classified in elementary aspects of event description, DOLCE foundational ontology and security domain concept model.

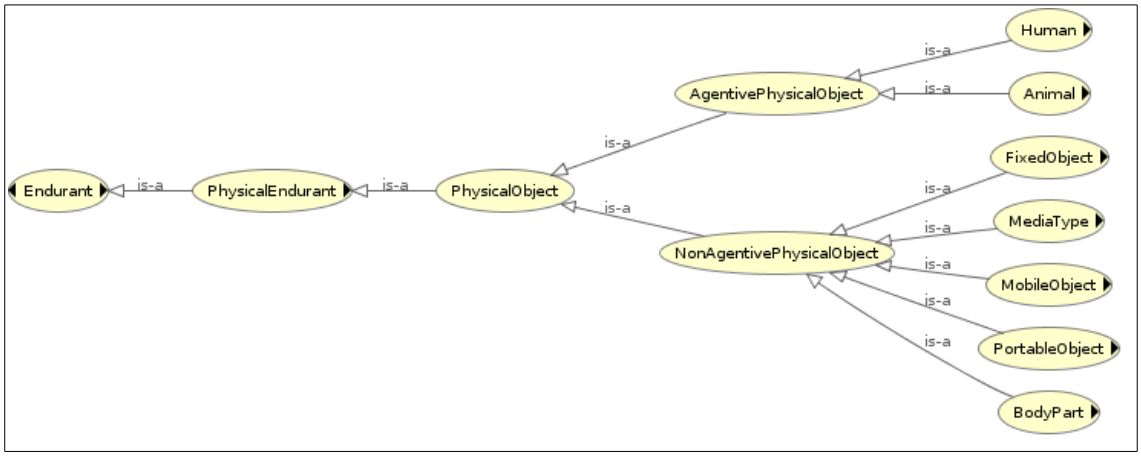


Figure 4.4: Physical object class hierarchies

Table 4.1: Informational aspect representation

Elementary Aspect	DOLCE		Security domain concepts		
Informational	Endurant – Physical Endurant – Physical Object	Agentive Physical Object	*Human*	*- Person* *- Police* *- Crowd*	
			Animal	*- Horse* *- Dog*	
		Non Agentive Physical Object	*Fixed Object*	*- Bank* *- Building* *- Hospital* *- Shop*	
			Mobile Object	*- Vehicle*	*- Car* *- Bus*
			Portable Object	*- Bin* *- Bottle* *- Pole* *- Battering Ram* *- Molotov Cocktail*	
			Body Part	*- Arm* *- Face* *- Head* *- Leg* *- Torso*	

Experiential aspect

A common event model that aims to serve as a base model for multimedia applications must show media awareness and let events refer to such media. Access to rich media documenting events is a natural prerequisite for this. Therefore, users should be offered engaging ways of exploring and experiencing a course of events to let them gain insights into how the events evolved. An event model's media referencing scheme should be capable of addressing media of different types, such as *images* or *videos* to any kind of sensor data that is available in the event. Therefore, **documentary support for events and objects** comprises the annotation of events and their participating objects with arbitrary information such as *media type* (representing types of content) and *media source* (the source of data from various devices).

Documentary support is also contributed by the vast usage of social media nowadays. Hence, *Internet* and *SocialNetwork* encompassed media source with information obtained from *Facebook*, *Twitter, blogs and forums* to name a few. Several social media concepts based on NLP are introduced in *PhysicalQuality* class to represent sentence components and POS tags in the ontology. The concepts include *Token*, *Sentence*, *Verb,* and *Noun*. In *PhysicalRegion* class, *StartEndNode, StartNode,* and *EndNode* concepts are defined to represent the character offsets in the source document for every individual populated through social media annotations. Figure 4.5 to 4.7 illustrate concepts that represent experiential aspects of event description and Table 4.2 elaborate the concepts.

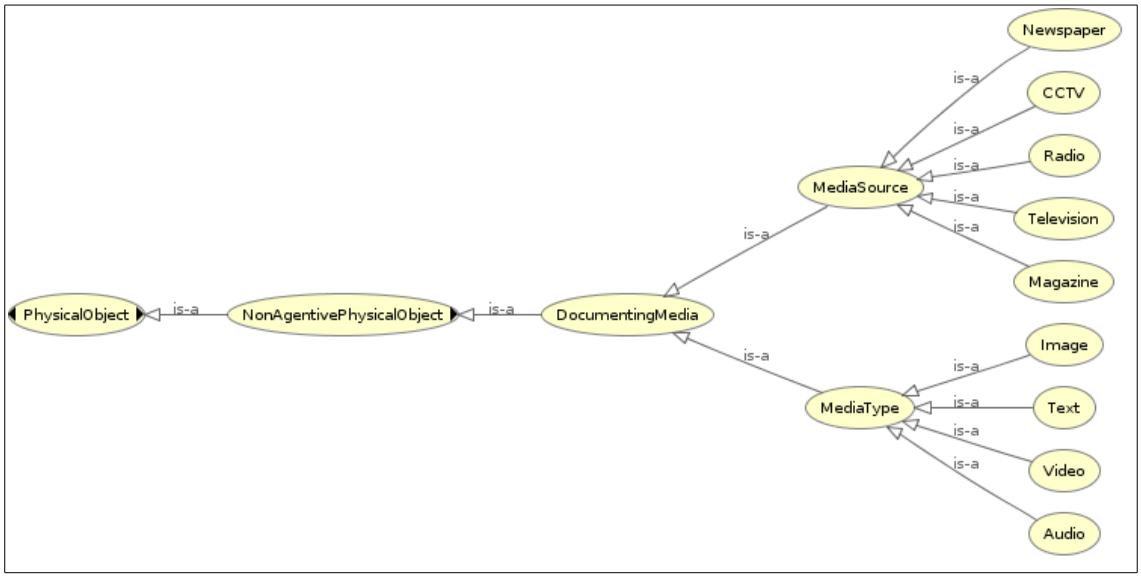


Figure 4.5: Documenting media class hierarchies

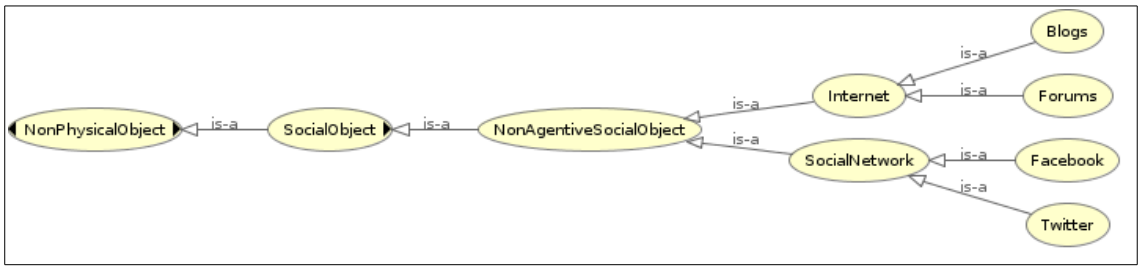


Figure 4.6: Media broadcast through social media sources

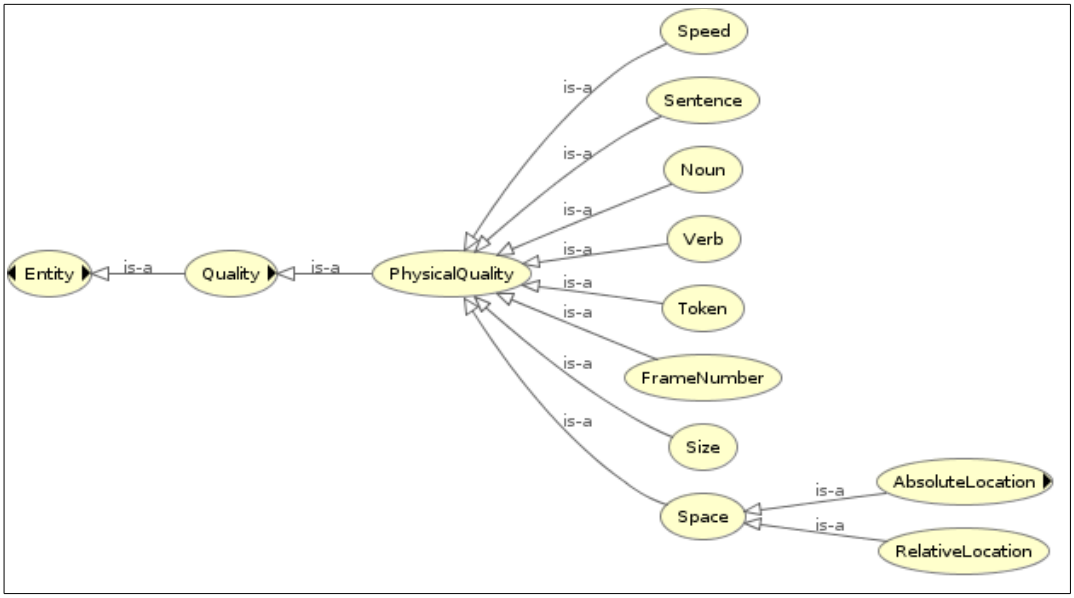


Figure 4.7: Physical quality concepts representing physical endurants

Table 4.2: Experiential aspect representation

Elementary Aspect	**DOLCE**		**Security domain concepts**		
Experiential	Endurant – Physical Endurant – Physical Object	Non Agentive Physical Object	*Documenting Media*	*Media Source*	*- CCTV* *- Radio* *- Television* *- Newspaper*
				Media Type	*- Image* *- Text* *- Video* *- Audio*
	Endurant – NonPhysical Endurant – NonPhysical Object – Social Object	Non Agentive Social Object		*Social Network*	*- Twitter* *- Facebook*
				Internet	*- Blogs* *- Forum*
	Quality	Physical Quality	*- Frame Number* *- Size* *- Speed* *- Token* *- Sentence* *- Verb* *- Noun*		

	Abstract – Quality Region	Physical Region	- *StartNode* - *EndNode* - *StartEnd Node*		

Spatial aspect

Objects unfold over space, thus modelling their spatial extension needs to be supported. An event model should show location awareness and support different ways of capturing the spatial aspect in an event's description. Different units of measurement, global, local, and media-related are needed to express the spatial aspect on a physical level. Therefore, in **spatial extension of objects,** location is modelled using *absolute location* and *relative location*. Figure 4.8 and Table 4.3 presents spatial aspect in security domain ontology.

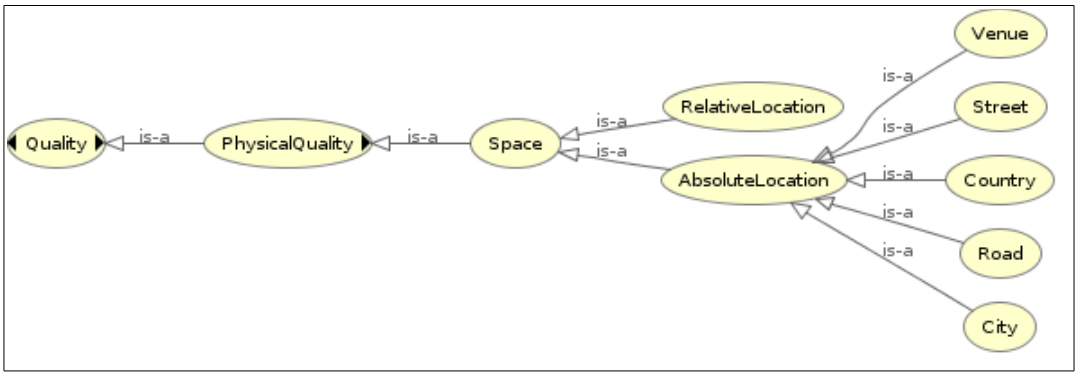


Figure 4.8: Location/Space class hierarchies

Table 4.3: Spatial aspect representation

Elementary Aspect	DOLCE		Security domain concepts	
Spatial	Quality – Physical Quality	Space	- *Absolute Location*	- *Country* - *City* - *Province* - *Region* - *Road* - *Street* - *Venue*
			- *Relative Location*	

Temporal aspect

As events unfold over time, their temporal duration needs to be modelled. The time of an event's occurrence is expressed in a global time measure, using a relative time measure, or in relation to media. Thus, **temporal duration of events** highlights concepts of *relative time* and *physical time* in temporal quality and temporal region concepts. Figure 4.9 and Table 4.4 presents the temporal aspect in security domain ontology.

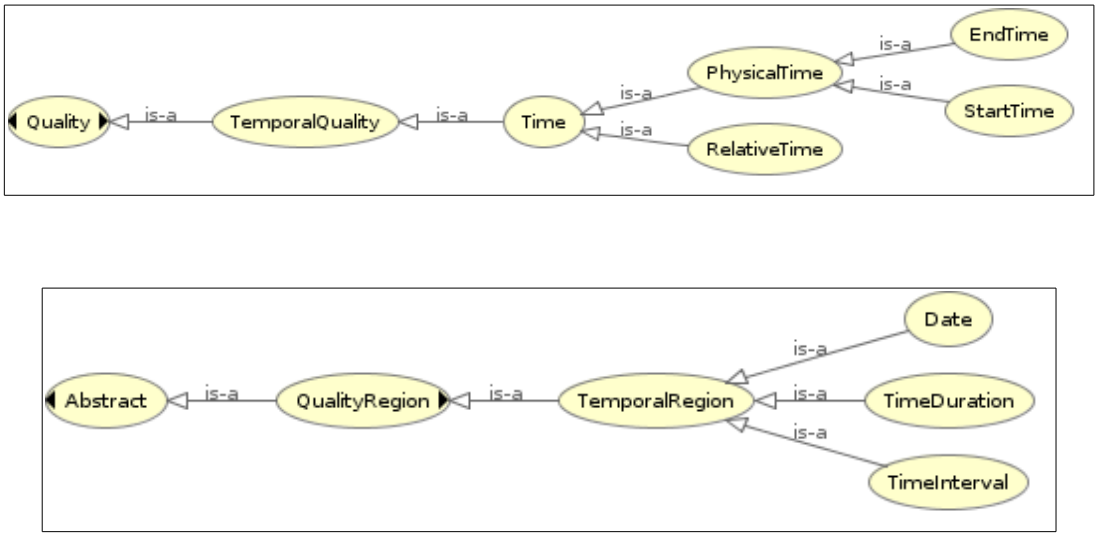


Figure 4.9: Time class hierarchies

Table 4.4: Temporal aspect representation

Elementary Aspect	DOLCE		Security domain concepts	
Temporal	Quality – Temporal Quality	Time	*- Physical Time*	*- StartTime* *- EndTime*
			- Relative Time	
	Abstract	Quality Region	*- Temporal Region*	*- Date* *- TimeDuration* *- Time Interval*

Structural aspect

Events are a modelling concept that is applicable at many different levels of abstraction. Thus, the exploration of the subevents that occurred as part of a more complex event offers important insights into an event's course. To address these, three kinds of structural relationships between events are being considered: (i) mereological, (ii) causal, and (iii) correlation relationships. On the other hand, the concept of perdurant (occurrence) in DOLCE foundational ontology are distinguished mainly on the basis of two notions: (1) homeomericity and (2) cumulativity.

An occurrence type is *Stative* or *Eventive* according to whether it holds the mereological sum of two of its instances. A sitting occurrence is stative because the cumulative sum of two sittings is still a sitting occurrence. Within stative occurrences, *States* and *Processes* are distinguished according to homeomericity. An occurrence is homeomeric if and only if all its temporal parts are described by the very expression used for the whole occurrence. As a further example, sitting is classified as a state, but running is classified as a process since there are (very short) temporal parts of a run that are not themselves runs. The concept *Pose* such as *Standing, Sitting, Laying* is categorized in states, and *Action* and *Gesture* such as *Walking, Running, Throwing, Kicking* and *Raise, StretchForward, MoveBackward* respectively, are categorize in process.

To define eventive occurrence concepts, criminal categories from Kaggle dataset [123] is used. Kaggle dataset contains London criminal reports of 33 Lower Super Output Areas (LSOA) boroughs, 7 major and 27 minor categories on monthly basis from January 2008 until December 2016. The major crime category includes *Burglary, CriminalDamage, SexualOffences, ViolenceAgainstthePerson, Robbery* and *TheftandHandling. T*hese categories are used to represent an *EventType* concept which represents *Accomplishment*

in the ontology. Finally, *Activities* namely *Fighting, Attacking* and *Looting* are defined in *Achievement* concept of the domain ontology.

Hierarchy of event type concepts is included to address the events correlation. Mereological relationship is represented between perdurant concepts as each concept represent different granularity degree which contributes to a higher level of concept representation. The concept of *Pose* represents the lowest granularity degree of action representation as it highlights stative action which has temporal parts that are unchanged for the whole occurrence. A combination of different *Pose, BodyParts* and Gesture create an *Action*. For example, the person who is ‘standing’ and ‘leg stretch forward’ is performing a ‘kicking’ action. Concepts represented in *Action* class represents an atomic action which are discrete actions that are carried out by a single person. Additionally, *Activity* concepts are composite actions which are composed of multiple atomic actions.

In the ontology, *Activity* concept represents a higher granularity degree of actions in event representation. Some examples are *Smashing,* represented by a sequence of *Walking* and *Hitting* action and *Attacking,* represented by a sequence of *Running* and *Throwing* action. Concepts defined in activity class also includes actions that represent an interaction between two people and which involves multiple atomic actions such as *Fighting.* Finally, *EventType* represents the highest granularity degree of event representation which consists and relies upon multiple concepts such as actions, activities, participants, objects, time and location. This hierarchical design enables working in different degrees of granularity and to decompose complex activity and event into simpler procedures. Figures 4.10, 4.11, 4.12 and 4.13 show class hierarchies representing perdurant entity and the concept of stative and eventive in security domain ontology and Table 4.5 summarizes the concepts.

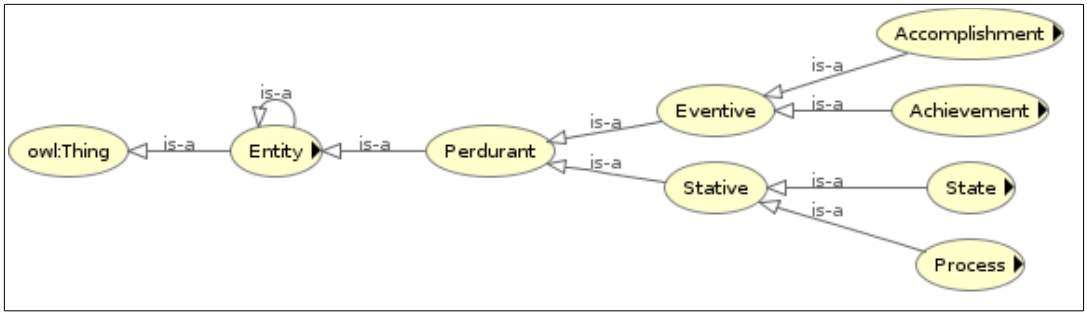


Figure 4.10: Class hierarchies representing Perdurant entities

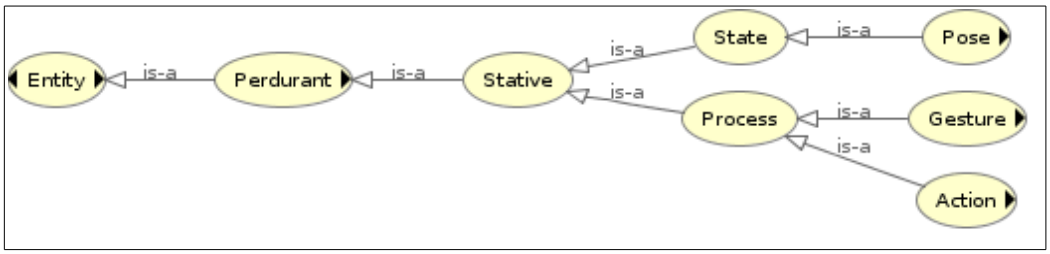


Figure 4.11: Class hierarchies representing the concept of Stative

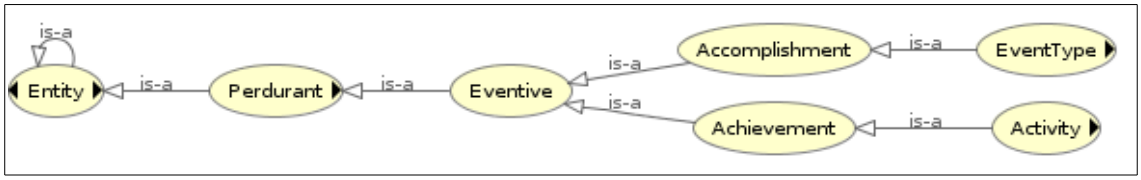


Figure 4.12: Class hierarchies representing the concept of Eventive

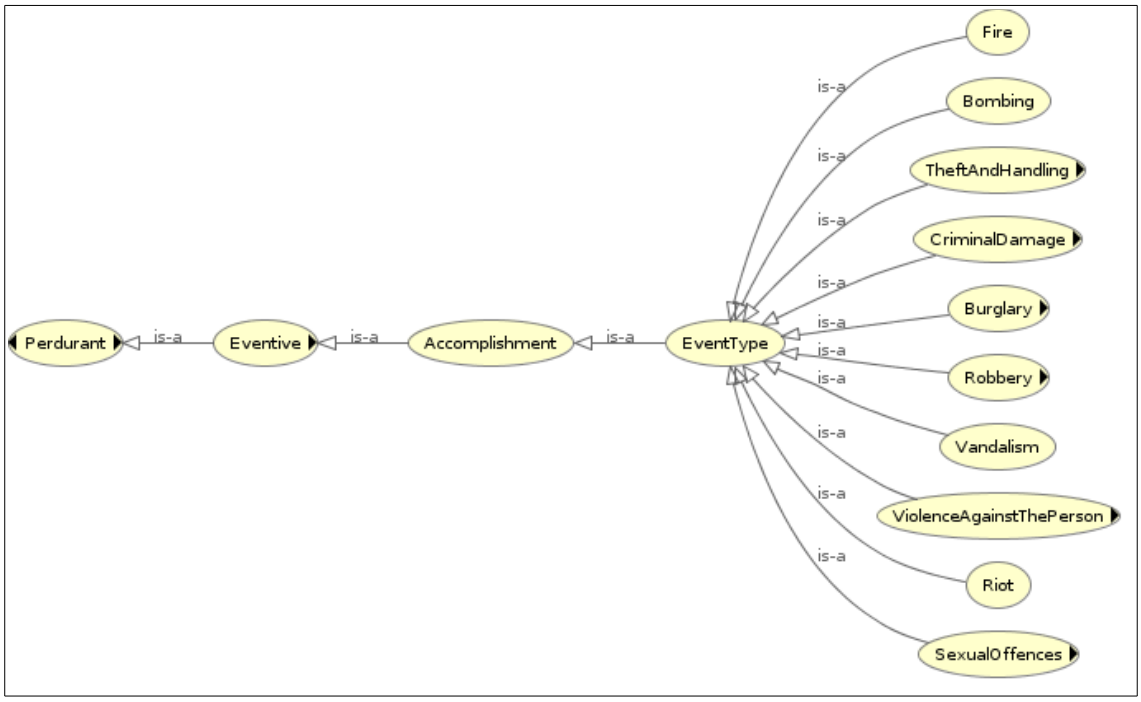


Figure 4.13: Class hierarchies representing the concept of EventType

Table 4.5: Structural aspect representation

Elementary Aspect	DOLCE		Security domain concepts	
Structural	Perdurant - Stative	State	*- Pose*	*- Laying* *- Sitting* *- Standing*
		Process	*- Gesture*	*- Lower* *- MovingBackward* *- MovingDown* *- MovingForward* *- MovingUp* *- Raise* *- StretchForward* *- StretchUpward* *- Swing* *- SwingForward* *- SwingLeftRight*
			- Action	*- Carrying* *- Hitting* *- Holding* *- Jumping* *- Kicking* *- Punching* *- Pushing* *- Running* *- Shooting* *- Throwing* *- Walking*
	Perdurant - Eventive	Achievement	*- Activity*	*- Attacking* *- Fighting* *- Looting* *- Smashing*
		Accomplishment	*- Event Type*	*- Bombing* *- Burglary* *- CriminalDamage* *- Fire* *- Riot* *- Robbery* *- Sexual Offences* *- Theft and Handling* *- Vandalism* *- Violence Against The Person*

Detail list from Kaggle dataset that represents event type which is grouped into different subclasses can be seen in Table 4.6.

Table 4.6: Criminal activity concepts from Kaggle dataset

	Event Type	
1.	*Burglary*	*Burglary in a dwelling*
		Burglary in other buildings
2.	*Criminal Damage*	*Criminal damage to dwelling*
		Criminal damage to motor vehicle
		Criminal damage to other building
		Other criminal damage
3.	*Robbery*	*Business property*
		Personal property
4.	*Sexual Offences*	*Rape*
		Other sexual
5.	*Theft and Handling*	*Handling stolen goods*
		Motor vehicle interference and tampering
		Theft from motor vehicle
		Theft from shops
		Theft/Taking of motor vehicle
		Theft/Taking of pedal cycle
		Other theft
6.	*Violence Against the Person*	*Assault with injury*
		Common assault
		Harrasment
		Murder
		Offensive weapon
		Wounding
		Other violence

Causal aspect

Relations between events such as causality and correlation can be a matter of subjectivity and interpretation (ambiguous or indistinct). Thus, the event model should offer means to express causality and permit the explicit representation of chains of causal events for individual events as shown in Table 4.7.

Table 4.7: Causal aspect representation

Elementary Aspect	DOLCE	Security domain concepts	
Causal	Perdurant – Eventive – Accomplishment – Event Type	*Event Causes*	*- Shooting* *- Protest*
		Event Effect	*- Looting* *- Robbery* *- Riot* *- Fire* *- Bombing* *- CriminalDamage* *- Violence Against the Person* *- Vandalism*

4.1.3.3 Semantic Relationship Between Concepts

Entities are the fundamental building blocks of OWL 2 ontologies, and they define the vocabulary (the named terms) of an ontology. Classes, datatypes, object properties, data properties, annotation properties, and named individuals are entities, and they are all uniquely identified by an IRI. *Classes* represent sets of individuals; *datatypes* are sets of literals such as strings or integers; *object* and *data properties* can be used to represent relationships in the domain; *annotation properties* can be used to associate nonlogical information with ontologies, axioms, and entities; and *named individuals* can be used to represent actual objects from the domain [124].

Relationships between entities specify how entities are related to other entities. Typically, a relation of a particular type (or class) specifies in what sense the object is related to the other object in the ontology. Much of the power of ontologies comes from the ability to describe relations. The set of relations describes the semantics of the domain. RDF is a language standard for representing ontologies which allow the definition of statements about things (or resources) in the form of RDF-triples or subject-predicate-object expressions.

Individual instances are the most specific concepts represented in an ontology. Individuals are created as class instances. Pairs of individuals are connected using *object properties. Data properties* connect individuals with literals. *Literals* represent data values such as particular strings or integers. This relationship allows more information to be included in the ontology structure.

4.1.3.4 Properties of Concepts

To address relationships between concepts in the domain, several concept properties are created to describe these relationships.

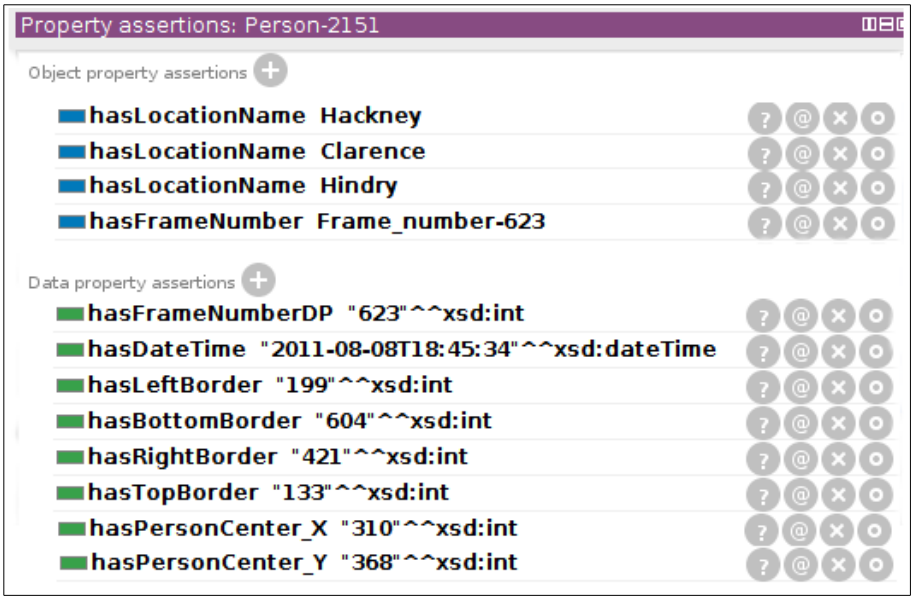


Figure 4.14: Property assertions for Person-2151

As seen in Figure 4.14, the object property *hasFrameNumber* is defined in the ontology to describe the frame number of detected features. For example, this figure shows that Person-2151 is detected in Frame_number-623 in the video footage. During the feature extraction process, every instance (detected feature) is asserted with a *hasFrameNumber* property so that the correlation between instances can be made based on frame number similarity in a video.

Instances are also assigned with bounding box information to represent its approximate location in the frame. For every instance, data property *hasLeftBorder, hasRightBorder, hasTopBorder, hasBottomBorder* is assigned to each side of the border. For face detection, data property *hasCenter_X* and *hasCenter_Y* is used to represent centre point coordinate of the circle. This property is used to recognize the location of a particular feature in the frame and correlation can be done with other features using similar properties. Figure 4.15 shows a correlation between an image from person detection process and another image form face detection process can be made by exploiting the frame number and bounding box information from both images.

Figure 4.15: Correlation between two images using instance properties

Additional object properties are created to describe relationships between concepts which are established through the reasoning process. As an example, an object property *hasFace* is created to describe the relationship between the concept of person and face and object property *isPerson* is created to describe the inverse functional characteristic of *hasFace* property. This example is illustrated in Figure 4.16.

'***Person-X hasFace Face-X***' and '***Face-X isPerson Person-X***'.

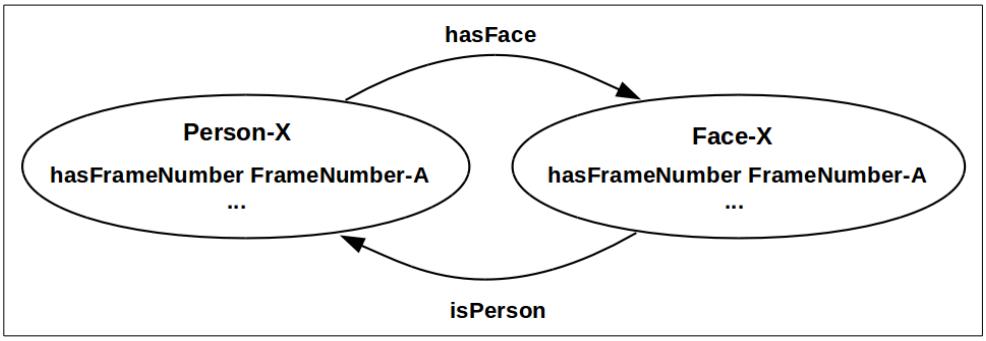


Figure 4.16: The relationship between Person-X and Face-X linked through object property 'hasFace' and 'isPerson'

Refering to Figure 4.14, *hasLocationName* object property is used to define the location property of events. In this study, location property is obtained from CCTV camera metadata's timestamp. Therefore, features extracted from video footage will be asserted with *hasLocationName* object property. Similarly, *hasDateTime* data property is assigned to every detected feature using metadata obtained from CCTV's timestamp. A list of object properties and data properties that is defined in the security domain ontology can be sobserved in Figure 4.17 and Table 4.8 lists domain and range for every object property.

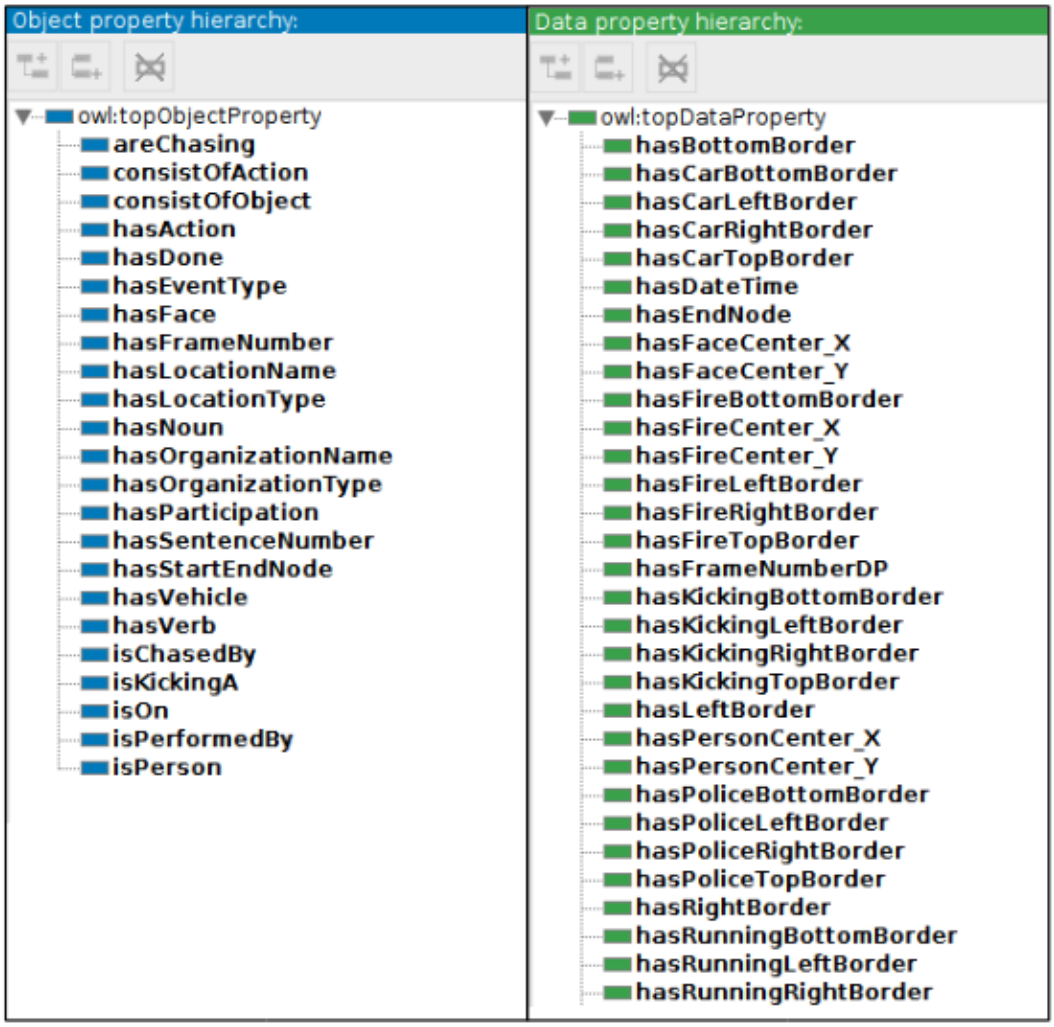


Figure 4.17: Object and data properties in security domain ontology

Table 4.8: Domain and range of object properties

Object Property	Domain	Range
hasFrameNumber	Human, Animal, Action, Gesture, Pose, BodyPart, MobileObject, FixedObject, PortableObject	FrameNumber
hasLocationName	Human, FixedObject, MobileObject, PortableObject	AbsoluteLocation
hasFace	Person	Face
isPerson	Face	Person
hasAction	Human	Action
isPerformedBy	Action, Gesture, Pose, EventType, Activity	Human

isKickingA	Person	MobileObject, FixedObject
hasDone	Person	EventType, Activity
isOn	FixedObject, MobileObject, PortableObject	Fire
hasEventType	Human, Activity, PortableObject, MobileObject, FixedObject, Action	EventType
areChasing	Police	Person
isChasedBy	Person	Police
consistOfAction	FrameNumber	Action
consistOfObject	FrameNumber	FixedObject, MobileObject, PortableObject
hasVehicle	FrameNumber	Vehicle
hasParticipant	FrameNumber	Human
hasLocationType	Token	AbsoluteLocation
hasOrganizationName	Token	Organization
hasOrganizationType	Token	Organization
hasNoun	Token	Noun
hasVerb	Token	Verb
hasSentenceNumber	Token	Sentence
hasStartEndNode	Token, Verb, Noun, Sentence, Organization, AbsoluteLocation	StartEndNode

4.1.4 Implementation

The security domain ontology concept hierarchy consists of 174 classes of enduring, perduring, abstract and quality entities, 23 object properties, 38 data properties and 490 axioms. Security domain ontology metrics are summarized in Table 4.9.

Table 4.9: Security domain ontology metric

Security Domain Ontology Metrics	
Axiom	490
Local axiom count	255
Declaration axioms count	235
Class count	174
Object property count	23
Data property count	38
Class axioms	
SubClassOf	173
Object property axioms	
InverseObjectProperties	2
ObjectPropertyDomain	53
ObjectPropertyRange	27

The ontology is developed based upon literature work on ontology engineering methodology and the concepts are identified and properly selected in conformity with elementary aspects of event description as discussed in [119]. For each requirement, the use case scenario is also explicitly referred to support wide circumstances and a variety of event conditions. The security domain ontology is also carefully aligned with the foundational ontology so that relevant semantic contents of the foundational ontology can be inherited with minimal effort. A complete illustration of security domain ontology that has been elaborated in this chapter can be seen in Appendix A. The ontology model is split into four parts for viewing clarity.

4.2. Summary

This chapter presents the development of an event conceptual model for security domain ontology. The development process implemented five phases of the ontology development methodology followed by refinement of the

ontology conceptual model in conformity with elementary aspects of the event description along with an adaptation of DOLCE foundational ontology for domain modelling. Validation of the complete system framework is conducted in the next chapter, beginning with visual semantic analysis in Chapter 5 and social media analysis in Chapter 6.

CHAPTER 5

VISUAL SEMANTIC ANALYSIS

This chapter presents experimental results for visual semantic analysis involving all modules of the system. The process begins with data acquisition from CCTV footage (introduced in Section 3.3); the implementation of automated and manual features extraction approach for low-level processing (demonstrated in Section 3.4.1); the parsing framework for inter-level data transformation (elaborated in Section 3.5); and the semantic reasoning and queries for knowledge retrieval (presented in Section 3.6). Both this thesis and research study done by Calavia et. al [10] implemented surveillance camera as sensor and semantic reasoning to perform semantic interpretation of the input data. Key features detections such as face and person, and manual features extractions were carried out using approaches proposed in [63][64] and [69]. Parsing framework approaches were studied in [77][84]. A complete visual semantic analysis framework is shown in Figure 5.1.

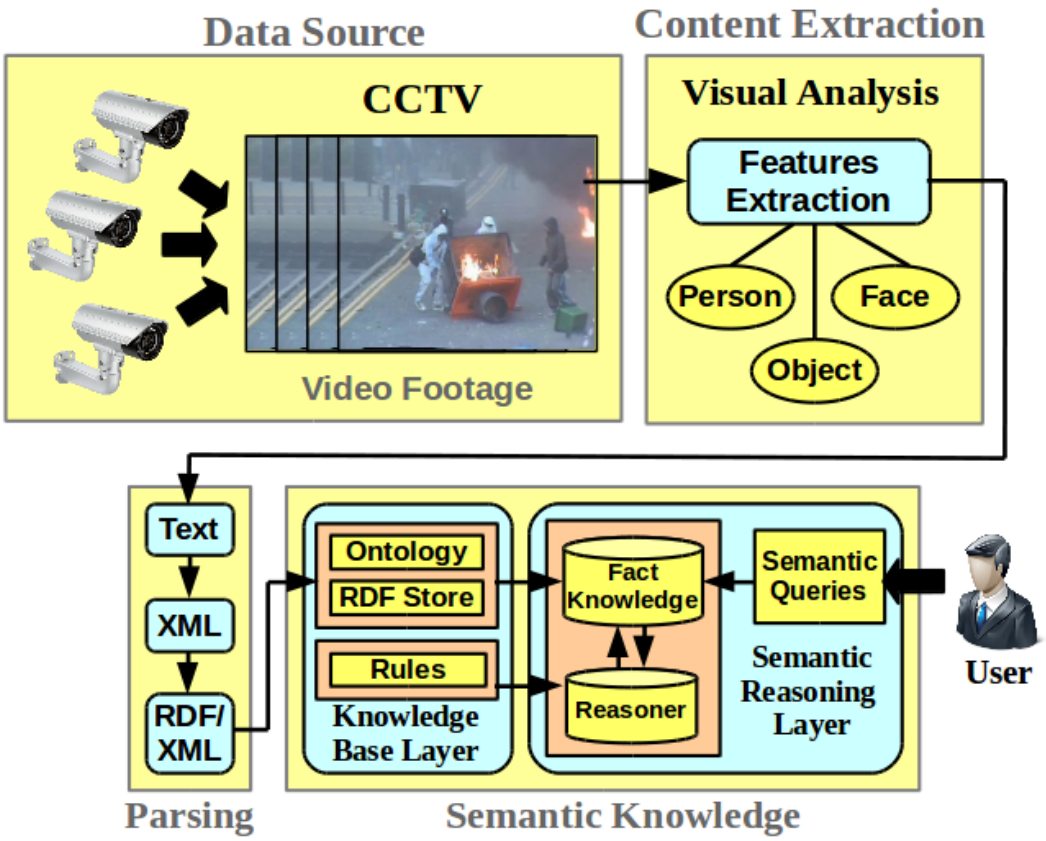


Figure 5.1: Visual semantic analysis framework

5.1. Dataset

The dataset for visual semantic analysis contains over 30 hours of video footage on the 2011 London Riots obtained from Scotland Yard. The video dataset was captured in various locations across London and represents diverse scenario including violent public disturbance behaviour, vandalism, and destruction of properties during the riots. The video dataset was carefully analysed and relevant videos are selected and used to validate the developed system framework. For visual semantic analysis, 10,855 frames of CCTV footage with resolution 704 x 625 pixels representing various situations in the event are used. Figure 5.2 shows example scenario during the 2011 London Riots.

Figure 5.2: The 2011 London Riots scenario

5.2. Visual Analysis

Visual analysis has been carried out to extract features that describe important concepts in the ontology. For event participant, the concepts *person*, *face,* and *police* are chosen since these play a main role during the event and are frequently encountered. The vehicle such as *car* and actions like *running, kicking and throwing* are also chosen as important concepts to be represented. The former is among the targeted object of vandalism and the latter represents recurrent actions during the riot. Since catastrophe related news is one of the most common topics that requires automatic retrieval [17], *fire* event is also chosen to be retrieved in the video.

5.2.1 Person Detection

The person detection approach executed on the video footage to detect people involved in the riot, is based on the HOG feature descriptor. The HOG person detector classifies *person* using a global feature consist of a dense grid of HOG descriptors tiled on the detection window. The HOG descriptor is computed for each position of the detector window and the classification process is done using SVM. Based on the video footage analysis, 34,125 instances of *person* are detected in the video. Although the video footage has a low resolution and people are not clearly visible in certain occasions in the video, this method successfully classifies *person* of various sizes under different conditions.

Open Computer Vision (OpenCV) library is used during video processing. Throughout the detection process, details such as the number of detected persons (person count), the corresponding frame number (frame number count), bounding box borders (using rectangle top-left coordinate, width and height), video capture timestamp and video timestamp (from CCTV metadata) are extracted and saved in a text file to record details of detected persons (refer Section 3.4.1.1). Example of successful person detections and a collection of saved images (on both sides) are shown in Figure 5.3. Figure 5.4 shows excerpt of recorded metadata for person detection process in Figure 5.3.

Figure 5.3: Example of people detection based on HOG person detector

```
Person-115.png Frame_number-29 29 Left: 521 Right: 611 Top: 12 Bottom: 199
PersonCenter_X: 566 PersonCenter_Y: 105 Position: 2315.97ms DateTime:
2011-08-08T18:45:34  Location: Hackney Street: Clarence/Hindry
Person-116.png Frame_number-29 29 Left: 429 Right: 539 Top: 20 Bottom: 268
PersonCenter_X: 484 PersonCenter_Y: 144 Position: 2315.97ms DateTime:
2011-08-08T18:45:34  Location: Hackney Street: Clarence/Hindry
Person-117.png Frame_number-29 29 Left: 12 Right: 112 Top: 17 Bottom: 269
PersonCenter_X: 62 PersonCenter_Y: 143 Position: 2315.97ms DateTime:
2011-08-08T18:45:34  Location: Hackney Street: Clarence/Hindry
Person-118.png Frame_number-29 29 Left: 551 Right: 681 Top: 84 Bottom: 376
PersonCenter_X: 616 PersonCenter_Y: 230 Position: 2315.97ms DateTime:
2011-08-08T18:45:34  Location: Hackney Street: Clarence/Hindry
Person-119.png Frame_number-29 29 Left: 146 Right: 264 Top: 35 Bottom: 300
PersonCenter_X: 205 PersonCenter_Y: 167 Position: 2315.97ms DateTime:
2011-08-08T18:45:34  Location: Hackney Street: Clarence/Hindry
Person-120.png Frame_number-29 29 Left: 228 Right: 382 Top: 55 Bottom: 402
PersonCenter_X: 305 PersonCenter_Y: 228 Position: 2315.97ms DateTime:
2011-08-08T18:45:34  Location: Hackney Street: Clarence/Hindry
```

Figure 5.4: Excerpt of recorded meta data from person detection process.

5.2.2 Face Detection

A Haar feature-based cascade classifier approach is used for face detection (refer Section 2.4.2 and Section 3.4.1.2). The implemented cascade classifier is able to detect different sizes of faces in the video, varying in the lighting and facial details. Face detection is performed to identify the people involved in the riot event. Based on the analysis, the Haar feature-based cascade classifier successfully classifies 12,408 instances of face.

Figure 5.5: Example of face detections using Haar feature-based cascade classifier

Similar to person detection, OpenCV library is used during the face detection process. Throughout the detection process, details such as the number of detected faces (face count), the corresponding frame number, centre point coordinates of the circle, video capture timestamp and video timestamp are extracted and saved in a text file to record details of detected faces (refer Section 3.4.1.2). An example of successful face detections and a collection of saved images (in the middle) are shown in Figure 5.5. Figure 5.6 shows excerpts of recorded metadata for face detection process in Figure 5.5 (left).

```
Face-1988.png Frame_number-1889 1889 Center: [243, 139] Position: 150857ms
DateTime: 2011-08-08T18:45:34  Location: Hackney Street: Clarence/Hindry
Face-1989.png Frame_number-1889 1889 Center: [416, 323] Position: 150857ms
DateTime: 2011-08-08T18:45:34  Location: Hackney Street: Clarence/Hindry
Face-1990.png Frame_number-1889 1889 Center: [225, 231] Position: 150857ms
DateTime: 2011-08-08T18:45:34  Location: Hackney Street: Clarence/Hindry
```

Figure 5.6: Excerpt of recorded meta data from face detection process in Figure 5.5 (left)

Since different methods of object detection are applied on the same video, the video capture timestamp helps to validate frame synchronization so that comparison between frames consisting object detection results can be made. As presented in Section 3.4.1.1, video capture timestamp indicates the time, in milliseconds (relative to the starting time) the object being detected after object detection algorithm has been executed. In Figure 5.7, video capture timestamp is marked by '*Position: 49753.4ms*' and both frames have the same timestamp. Therefore, it can be concluded that these two frames are the same frame in the video footage and objects that are detected in that frame using different detection methods can be matched.

Figure 5.7: Video capture timestamp extracted during video processing

The matching process can be done by utilizing bounding box information. For every person detected, the bounding box carries information of left, right, top and bottom border indicating the location of the person in the frame. For face detection, the centre point coordinates of the detected face are marked. Using semantic rules, the face is matched to a person if the person and the face existed in the same frame (or has the same video capture timestamp) and the centre point coordinates of the face are located within the border of the person.

Correlations can also be done with other features using similar properties. The methods implemented in the visual analysis module are chosen because they provide good real-time detection results with lower computational costs despite both not being the most recent methods.

5.2.3 Action Recognition and Object Detection

For action recognition, instances of *running*, *throwing* and *kicking* are manually annotated in the video using the ViPER-GT annotation tool as presented in Section 3.4.1.3. Apart from that, object annotation has also been performed on *car*, *police* and *fire* during the event. Figure 5.8 shows examples of *running, throwing* and *kicking* action annotations and *car*, *police,* and *fire* object annotations performed on the video footage of the riot event.

Figure 5.8: Actions and objects annotation from CCTV footage

5.3. Parsing Framework Implementation

The parsing process is implemented to bridge the gap between different data representation (Text-XML-OWL) in the system. The sequence of parsing process has been shown in Figure 3.9, Section 3.5.2 in Chapter 3. This transformation can be simplified in three phases by breaking it down. The first phase produces XML that is isomorphic to the original text. The second phase restructures the XML into RDF/XML, and the third phase merges the newly generated RDF/XML with the existing OWL ontology model. The next subsection explains the detailed textual to OWL transformation process.

5.3.1 Textual File Generation

Prior to the parsing phase, the visual analysis module processes video data recorded by the surveillance camera and produced a list of detected objects' metadata in a text file. For instance, person detection algorithms implemented in OpenCV are programmed to generate details as can be seen in Figure 5.9.

```
Person-1.png Frame_number-2 2 Left: 418 Right: 480 Top: 98 Bottom: 238 Position:
159.722ms DateTime: 2011-08-08T18:45:34  Location: Hackney Street: Clarence/Hindry
Person-2.png Frame_number-2 2 Left: 639 Right: 696 Top: 49 Bottom: 176 Position:
159.722ms DateTime: 2011-08-08T18:45:34  Location: Hackney Street: Clarence/Hindry
Person-3.png Frame_number-2 2 Left: 351 Right: 408 Top: 96 Bottom: 223 Position:
159.722ms DateTime: 2011-08-08T18:45:34  Location: Hackney Street: Clarence/Hindry
Person-4.png Frame_number-2 2 Left: 231 Right: 285 Top: 124 Bottom: 247 Position:
159.722ms DateTime: 2011-08-08T18:45:34  Location: Hackney Street: Clarence/Hindry
Person-5.png Frame_number-3 3 Left: 637 Right: 696 Top: 56 Bottom: 189 Position:
239.583ms DateTime: 2011-08-08T18:45:34  Location: Hackney Street: Clarence/Hindry
Person-6.png Frame_number-3 3 Left: 327 Right: 386 Top: 103 Bottom: 236 Position:
239.583ms DateTime: 2011-08-08T18:45:34  Location: Hackney Street: Clarence/Hindry
Person-7.png Frame_number-3 3 Left: 494 Right: 548 Top: 134 Bottom: 255 Position:
239.583ms DateTime: 2011-08-08T18:45:34  Location: Hackney Street: Clarence/Hindry
Person-8.png Frame_number-4 4 Left: 638 Right: 696 Top: 56 Bottom: 201 Position:
319.444ms DateTime: 2011-08-08T18:45:34  Location: Hackney Street: Clarence/Hindry
Person-9.png Frame_number-4 4 Left: 302 Right: 363 Top: 113 Bottom: 249 Position:
319.444ms DateTime: 2011-08-08T18:45:34  Location: Hackney Street: Clarence/Hindry
Person-10.png Frame_number-4 4 Left: 485 Right: 539 Top: 148 Bottom: 269 Position:
319.444ms DateTime: 2011-08-08T18:45:34  Location: Hackney Street: Clarence/Hindry
```

Figure 5.9: Excerpt from person detection metadata file

The first column is a label of the detected person in the video. Every detected person is given a number starting from *Person-1* until the final detected person in the footage. The second column represents the frame number in which the

person detection was made in the video. Frame number repetition can be seen in the list as multiple person detections are made in each frame. Four columns to the right (left, right, top and bottom) describe four points that represent bounding boxes' borders of the detected person in the video. These points represent the location of the detected person in the frame providing useful information to locate the person and perform a comparison between the detected object in different frames. Next is the video capture timestamp which indicates the time the person being detected after the algorithm has been executed. Finally, the *DateTime*, *Location,* and *Street* are extracted information obtained from the surveillance camera metadata.

Alternatively, manual annotation executed using the ViPER-GT tool generates an XML-based file format to define and instantiates descriptors based on the annotation task. The descriptor is composed of descriptor type, descriptor name, attribute type, attribute name and its instances. The descriptor attribute with type bbox provides the coordinates of the top-left corner of the box, the height, the width and the annotation frame span of the object in the video footage. From this information, a similar text file format as previously explained in Section 5.3.1 is produced. An example of descriptor file for *throwing* action generated by ViPER-GT annotation tool is shown in Figure 5.10.

```
<object framespan="1848:1861" id="22" name="AtomicAction">
  <attribute name="Throwing">
     <data:bbox framespan="1848:1848" height="249" width="156" x="237" y="319"/>
     <data:bbox framespan="1849:1849" height="249" width="139" x="253" y="319"/>
     <data:bbox framespan="1850:1850" height="249" width="127" x="264" y="319"/>
     <data:bbox framespan="1851:1851" height="228" width="127" x="264" y="339"/>
     <data:bbox framespan="1852:1852" height="228" width="127" x="284" y="341"/>
     <data:bbox framespan="1853:1853" height="228" width="138" x="272" y="341"/>
     <data:bbox framespan="1854:1854" height="228" width="158" x="272" y="341"/>
     <data:bbox framespan="1855:1855" height="291" width="167" x="250" y="278"/>
     <data:bbox framespan="1856:1856" height="255" width="154" x="277" y="312"/>
     <data:bbox framespan="1857:1857" height="255" width="175" x="281" y="317"/>
     <data:bbox framespan="1858:1858" height="225" width="175" x="297" y="347"/>
     <data:bbox framespan="1859:1859" height="220" width="175" x="297" y="351"/>
     <data:bbox framespan="1860:1862" height="220" width="167" x="297" y="351"/>
  </attribute>
</object>
```

Figure 5.10: 'Throwing' in XML generated by ViPER-GT annotation tool

5.3.2 Text to XML Data Conversion

Utilizing the generated text file, an XML file is created from the metadata available in the text file using SAX Parser. In this phase, a hierarchical tree structure is created with defined classes and subclasses so that instances can be mapped to a correct XPath location in the ontology. As presented in Section 2.5.1.1, XPath is used to address nodes in an XML document. For example, the XPath to address *Person-1* in Figure 5.11 is */Entity/Endurant/PhysicalEndurant/PhysicalObject/AgentivePhysicalObject/Human/Person/NamedIndividual/.*

```
<Entity>
  <Endurant>
    <PhysicalEndurant>
      <PhysicalObject>
        <AgentivePhysicalObject>
          <Human>
            <Person>
              <NamedIndividual>Person-1</NamedIndividual>
              <hasFrameNumber>Frame_number-2</hasFrameNumber>
              <hasFrameNumber>2</hasFrameNumber>
              <hasLeftBorder>418</hasLeftBorder>
              <hasRightBorder>480</hasRightBorder>
              <hasTopBorder>98</hasTopBorder>
              <hasBottomBorder>238</hasBottomBorder>
              <hasDateTime>2011-08-08T18:45:34</hasDateTime>
              <hasLocationName>Hackney</hasLocationName>
              <hasLocationName>Clarence</hasLocationName>
              <hasLocationName>Hindry</hasLocationName>
            </Person>
            …
            …
          </Human>
        </AgentivePhysicalObject>
      </PhysicalObject>
    </PhysicalEndurant>
  </Endurant>
</Entity>
```

Figure 5.11: Excerpt of Person.xml

The XML instance data consist of several root elements with multiple nodes in the lowest child element. The lowest child elements in the XML file have unique descriptions between one another and represent names and properties of every instance. Figure 5.11 shows an excerpt of the Person.xml file after being parsed from a text file using SAX Parser. The XML document contains the instance *Person-1* with a list of attributes obtained during the object detection process.

During the Text to XML parsing process, *Person-X* is tagged as *NamedIndividual, Frame_number-X is tagged as hasFrameNumber,* four bounding box borders are tagged as *hasLeftBorder, hasRightBorder, hasTopBorder and has BottomBorder,* date-time timestamp and location are tagged as *hasDateTime* and *hasLocationName. Person* instances are assigned in *Person* class which is a subclass of *Entity–Endurant–PhysicalEndurant–PhysicalObject–AgentivePhysicalObject–Human*, as can be seen in the class hierarchy. An *Instance-Class* assignment is executed individually referring to the type of features that is annotated.

5.3.3 XML to RDF/XML Mapping and Ontology Population

The generated XML document is restructured into an RDF/XML document, using an XSLT stylesheet. This transformation interprets the tree structure of XML and represents the intended model in the RDF/RDFS subject-predicate-object structure. A produced XSLT stylesheet can be used by any XSLT processor to automatically generate the desired ontology. During the transformation, every node is mapped according to OWL ontology concept for a successful integration between newly generated XML instances and the existing OWL. An excerpt of the XSLT stylesheet that has been created is shown in Figure 5.12.

The process of executing the XSLT stylesheet over the XML instances document, produces an output document containing all the generated individuals plus their properties in RDF/XML format. The RDF/XML instances are merged with an existing OWL ontology to which the prefix class is bound. The newly generated OWL ontology can be loaded in any OWL editor such as Protégé. Figure 5.13 shows the ontology population of *Person* instances with generated object properties and data properties for every Person instance in Protégé.

```
<!-- Individuals -->
   <xsl:for-each select="Entity/Endurant/Substantial/PhysicalSubstantial/PhysicalObject/
                         AgentivePhysicalObject/Human/Person">
   <xsl:variable name="IndividualDetails" select="."/>
      <xsl:element name="owl:NamedIndividual">

         <xsl:attribute name="rdf:about">
         <xsl:attribute name="xml:base">http://www.semanticweb.org/farhan/ontologies/ForensicDomain#
         </xsl:attribute>
            <xsl:value-of select="$IndividualDetails/NamedIndividual"></xsl:value-of>
         </xsl:attribute>

            <xsl:element name="rdf:type">
               <xsl:attribute name="rdf:resource">
               <xsl:attribute name="xml:base">http://www.semanticweb.org/farhan/ontologies/ForensicDomain#
               </xsl:attribute>
                  <xsl:value-of select="$EighthElementName"></xsl:value-of>
               </xsl:attribute>
            </xsl:element>

            <xsl:element name="ForensicDomain:hasFrameNumber">
               <xsl:attribute name="rdf:resource">
               <xsl:attribute name="xml:base">http://www.semanticweb.org/farhan/ontologies/ForensicDomain#
               </xsl:attribute>
                  <xsl:value-of select="$IndividualDetails/hasFrameNumber"></xsl:value-of>
               </xsl:attribute>
            </xsl:element>

            <xsl:element name="ForensicDomain:hasLeftBorder">
               <xsl:attribute name="rdf:datatype">http://www.w3.org/2001/XMLSchema#int</xsl:attribute>
               <xsl:attribute name="xml:base">http://www.semanticweb.org/farhan/ontologies/ForensicDomain#
               </xsl:attribute>
                  <xsl:value-of select="$IndividualDetails/hasLeftBorder"></xsl:value-of>
            </xsl:element>
            ...
            ...
      </xsl:element>
   </xsl:for-each>
```

Figure 5.12: Excerpt of XSLT stylesheet for XML to RDF/XML transformation

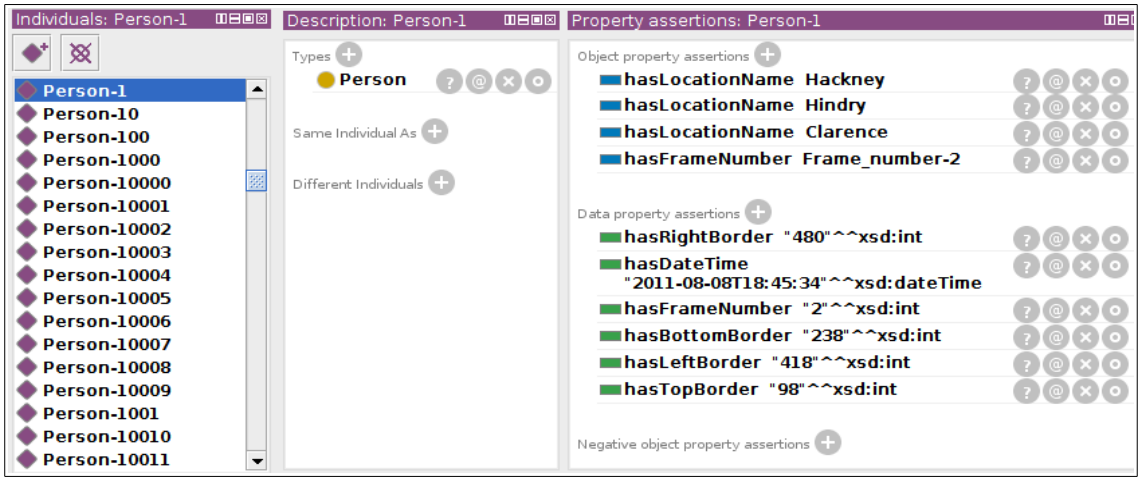


Figure 5.13: Ontology population of Person instances and its properties

5.4. Formal Representation of Event

The formal representation of events is demonstrated in this section, to show the implementation of SWRL rules, semantic reasoner, and queries on the knowledge base for event understanding.

5.4.1 Examples of Scenario

The event representation is based on several examples of scenario during the riot event that gives rise to a number of situations. The description is presented in a natural language before a formal language representation is made. Several scenarios of interest that are going to be represented were identified in the video footage. As an example, one scene in the video footage shows that people were running and throwing objects at the police. Another scene shows rioters kicking and damaging a car parked at the roadside. There is also video footage showing rioters that were chased by the police. Based on these three descriptions, objects and actions involved in these situations are identified and formal event representations are formed.

The key advantage of having situations represented in a formal language are that facts of knowledge not explicitly stated in the knowledge base can be derived using an inference engine. For instance, an *attack* activity can be derived when two consecutive atomic actions (*running* and *throwing)* are performed almost simultaneously. An event of a burning *car* can be derived when an object *car* is detected at the same location and frame number in which a *fire* is detected and a *chasing* event can be derived when two opponents are *running* in the same direction. Figure 5.14 shows the scenario of events that is captured in CCTV footage of the riot event.

Figure 5.14: Event scenario captured from CCTV footage of a riot event

5.4.2 Representation of Situations

Examples of situations described in the previous section are captured in a formal language, predominately described by objects such as *person*, *face*, *car*, *fire* and *police,* and actions which are *running*, *throwing* and *kicking.* A number of classes, properties, and relations are introduced to represent these situations (Section 4.1.3). This section will present how the situations are represented in a formal language.

i. Representation of **‘*PersonhasFrameNumberFrameNumber*’**

FrameNumber is a concept created to represent the frame number of detected instance in visual analysis module. All concept instances are linked to frame number using the *‘hasFrameNumber’* object property. As an example, *‘(Object)hasFrameNumberFrameNumber’* represent a link of an object with its corresponding frame number and *‘(Action)hasFrameNumberFrameNumber’* represent a link of an action with the corresponding frame number which the action is detected in the video footage.

ii. Representation of ‘**PersonhasFaceFace’**

This representation links an agentive physical object, a *person* in the first parameter and non-agentive physical object, a *face* in the second parameter using *‘hasFace’* object property. The object property *‘hasFace’* is an inverse functional property of *‘isPerson’* where a *face* is matched with a *person* using *‘FaceisPersonPerson’.* This match is made by utilizing frame number similarity and location of detected person and face in the frame. Since both person and face detection is being conducted using different object detection methods, an inference approach is used to match these concepts together by using semantic rules.

iii. Representation of **‘*PersonhasActionRunning*’**

Running is classified in the *Action* class which represents the lowest granularity degree of action representation. In this statement, a relation is created between two concepts: object in the first parameter followed by action in the second parameter. Both concepts are linked by the *‘hasAction’* object property. *‘RunningisPerformedByPerson’* is an inversed representation of *‘PersonhasActionRunning’* which introduces *‘isPerformedBy’* object property, an inverse functional property of *‘hasAction’*. The attribute of this situation includes the location of the situation, date-time timestamp associated with the occurrence of the action, the frame number properties of both concepts and bounding box borders of person and running action. A similar representation can be used to represent other action such as *‘PersonhasActionKicking’.*

iv. Representation of ***‘PersonhasActionAttacking’***

Attacking is defined in the *Activity* class*,* which represents a composite action derived from two consecutive atomic actions, *running* and *throwing.* The first parameter is represented by an object and the second parameter is an activity, linked by the *‘hasAction’* object property. *Attacking* activity is derived through inference process with the support of semantic rules. The attribute of this situation includes the location and date-time timestamp associated with the occurrence of the action and the material object involved in the action. The sub-action *running* and *throwing* have attributes of their own, for example, speed and velocity of running action while performing the attack. This higher-level action and activity inference is used to represent situations such as *‘PersonhasActionSmashing’,* and ‘*PersonhasActionFighting*’. However, derivation of these activities *(smashing and fighting)* are not presented in this thesis.

v. Representation of **‘*PersonisKickingACar*’**

In this statement, three concepts are linked together in a single statement. AgentivePhysicalObject *(person)*, action *(kicking)* and material object *(car*) are linked by the *‘isKickingA’* object property. Objects and actions performed on the scene are formulated using semantic rules to deduce this event. The attribute for this situation includes time and location in which this event takes place. Another related statement that can be associated with this event is *‘PersonhasDoneVandalism’* since kicking a car is an act of vandalism.

vi. Representation of ***‘PoliceareChasingPerson’***

This statement represents a chasing activity which happens between *police* as can be seen in the first parameter and *person* indicated by the second parameter linked by the *‘areChasing’* object property. This event is deduced from two events that occur simultaneously: ‘*PolicehasActionRunning’* and ‘*PersonhasActionRunning’.* Chasing is defined as a pursuit after something in order to catch. Thus, *running* is the key action that represents this activity. The object property *‘areChasedBy’* is defined as an inverse functional property of *‘areChasing’* and can be represented in the statement *‘PersonareChasedByPolice’.* The attribute for this situation includes the time and location in which this event takes place.

vii. Representation of ***‘CarisOnFire’***

This statement links the mobile object ‘C*ar’* and event type *‘Fire’* using a semantic rule and inference approach. The object and event type both need to be located in the same frame number and around the same location in the image frame. Another statement that can be associated with this event is *‘EventTypeisCriminalDamage’* considering the act of damaging personal property is a criminal damage.

5.5. Semantic Reasoning

5.5.1 Rule-based Classification

A set of rules is proposed in this study to model the situations and deduce new knowledge in the ontology. As presented in Section 5.2.2, every instance in visual analysis module were asserted with instance properties. By exploiting these asserted knowledge, new relations between instances were derived using rule-based inference process. The Horn clause rules were implemented in this study because these rules are more general, applied to arbitrary concept instances and are much more succinct [125]. These rules were implemented using SWRL supported by the Pellet reasoner [109] (refer Section 3.6.3).

5.5.2 Rules Implementation

The knowledge integration is made using semantic rules that chain together asserted data. Five rules have been created to demonstrate semantic reasoning and knowledge inference in the ontology. For example, utilizing frame number information, correlation between the suspect's face and performed actions could be established although detection of both features is being made separately. The Protégé rule plugin has been used to write the inference rules in SWRL language.

Rule 1: Relationship between *Person* and *Face*

Rule 1:	**Person-Face**
Inferred knowledge:	**PersonhasFaceFace/FaceisPersonPerson**
Person(?A),Face(?B),FrameNumber(?C),hasFrameNumber(?A,?C), hasFrameNumber(?B,?C),hasLeftBorder(?A,?W),hasRightBorder (?A,?X),hasTopBorder(?A,?Y),hasBottomBorder(?A,?Z), hasFaceCenter_X(?B,?M),hasFaceCenter_Y(?B,?N),greaterThan (?M,?W),lessThan(?M,?X),greaterThan(?N,?Y),lessThan(?N,?Z) -> hasFace(?A,?B),isPerson(?B,?A)	

Rule 1 is constructed as follows: The reasoner checks and compares the frame number of a person's and face's instances in the current frame. The first condition to fulfil this rule is that both instances should exist in the same frame. The reasoner is then supplied with the person's bounding box borders and the face's centre point coordinates. The process of comparing the centre point coordinate and the bounding box borders is then executed. If the face's centre point coordinate is located within the person's bounding box borders, the reasoner will infer that the person has the corresponding face instance and the face has the corresponding person instance.

Rule 2: Relationship between *Person* and Actions *(Running/Throwing/Kicking)*

Rule 2:	**Person – Running/Throwing/Kicking**
Inferred knowledge:	**PersonhasActionRunning/ RunningisPerformedByPerson**
Person(?A),Running(?B),FrameNumber(?C),hasFrameNumber (?A,?C),hasFrameNumber(?B,?C),hasPersonCenter_X(?A,?W), hasPersonCenter_Y(?A,?X),hasRunningLeftBorder(?C,?O), hasRunningRightBorder(?C,?P),hasRunningTopBorder(?C,?Q), hasRunningBottomBorder(?C,?R),greaterThan(?W,?O), lessThan(?W,?P),greaterThan(?X,?Q),lessThan(?X,?R) -> hasAction(?A,?B),isPerformedBy(?B,?A)	

Rule 2 presented above is constructed as follows: The reasoner checks and compares the frame number of person and action instances in the current frame. The first condition to fulfil this rule is that both instances should exist in the same frame. The reasoner is then supplied with person's bounding box centre point coordinates and action's bounding box borders and checks if the centre point coordinate is located within the action's bounding box borders. If the condition is true, the reasoner will infer that the person has performed the particular action. *Running* action is being demonstrated in this example. However, this rule can also be used to infer other action instances.

Rule 3: Relationship between *Person* and Action *(Attacking)*

Rule 3:	**Person – Attacking**
Inferred knowledge:	**PersonhasActionAttacking/ AttackingisPerformedByPerson**
Person(?A),Running(?C),hasAction(?A,?C),Throwing(?D), hasAction(?A,?D),Attacking(?F) -> hasAction(?A,?F), isPerformedBy(?F,?A)	

Rule 3 needs to be implemented together with Rule 2 since Rule 2 provides information about the action that a person has performed. Based on this information, the reasoner will check and compare the frame number of the person and every action instance done by the person, and infer an *attack* activity if a person performs both *running* and *throwing* actions in the video footage.

Rule 4: Relationship between *Person,* Action *(Kicking)* and Object *(Car)*

Rule 4:	**Person – Kicking – Car/Person - Vandalism**
Inferred knowledge:	**PersonisKickingACar/PersonhasDoneVandalism**
Person(?A),Kicking(?B),Vehicle(?M),FrameNumber(?C), hasFrameNumber(?A,?C),hasFrameNumber(?B,?C), hasFrameNumber(?M,?C),hasPersonCenter_X(?A,?W), hasPersonCenter_Y(?A,?X),hasKickingLeftBorder(?C,?O), hasKickingRightBorder(?C,?P),hasKickingTopBorder(?C,?Q), hasKickingBottomBorder(?C,?R),greaterThan(?W,?O), lessThan(?W,?P),greaterThan(?X,?Q),lessThan(?X,?R), Vandalism(?V) -> isKickingA(?A,?M),hasDone(?A,?V)	

In Rule 4, the reasoner checks and compares the frame number of the person, kicking and car instances in the current frame. The first condition to fulfil this rule is that all three types of instances should exist in the same frame. The reasoner is then supplied with person's bounding box centre point coordinates

and action's bounding box borders and checks if the centre point coordinate is located within the action's bounding box borders. If the condition is true, the reasoner will infer that the person has performed the particular action. In this example, Rule 4 will check if the person has performed a *Kicking* action and identify any *Vehicle (Car)* availability in the same frame. If a vehicle is available, the rule will infer a new knowledge *PersonisKickingACar.* This rule also infers a knowledge *PersonhasDoneVandalism* to describe the event.

Rule 5: Object *(Car)* and Event Type *(Fire)*

Rule 5:	**Car – Fire/EventType - CriminalDamage**
Inferred knowledge:	**CarisOnFire/EventTypeisCriminalDamage**
Vehicle(?A),Fire(?B),FrameNumber(?C),hasFrameNumber (?A,?C),hasFrameNumber(?B,?C),hasFireCenter_X(?B,?W), hasFireCenter_Y(?B,?X),hasVehicleLeftBorder(?C,?O), hasVehicleRightBorder(?C,?P),hasVehicleTopBorder(?C,?Q), hasVehicleBottomBorder(?C,?R),greaterThan(?W,?O), lessThan(?W,?P),greaterThan(?X,?Q),lessThan(?X,?R), EventType(?T),CriminalDamage(?U) -> isOn(?A,?B),is(?T,?B), is(?T,?U)	

In Rule 5, the reasoner checks and compares the frame number of vehicle (car) and fire instances in the current frame. The first condition to fulfil this rule is that both types of instances should exist in the same frame. The reasoner is then supplied with fire's bounding box centre point coordinates and vehicle's bounding box borders and checks if the fire's centre point coordinate is located within the vehicle's bounding box borders. If the condition is true, the reasoner will infer that the *CarisOnFire* and this event is also inferred as *EventTypeisCriminalDamage.*

5.6. Knowledge Retrieval

5.6.1 Querying Formal Representations

The difference between queries to a database and queries to an OWL knowledge base is that the answer to a knowledge base query includes facts that are inferred as well as facts that have been explicitly asserted. SPARQL [126] provides a formal language to ask meaning-driven questions in the knowledge base and is used to express these queries.

Table 5.1 presents SPARQL queries for SWRL rules that have been presented in Section 5.5.2, followed by query results from the knowledge base in the consecutive section. The queries are able to extract various information such as ‘Who is running?’, ‘Who is attacking?’, ‘What are the actions detected in the video?’, ‘What actions are being done by Person-X?’, ‘Which frame number does the burning car detected?’ and etc. The results demonstrate how ontology-based reasoning and queries are utilized to extract meaningful information which leads to *suspect, action* and *event* identification from the video footage.

Table 5.1. SWRL rules, Snap SPARQL queries and query results

Query 1: Person - Face
SELECT ?ind ?Face ?hasFrameNumber **WHERE {** **?ind rdf:type SecurityDomain:Person .** **?ind SecurityDomain:hasFace ?Face .** **?ind SecurityDomain:hasFrameNumber ?hasFrameNumber .** **}** **ORDER BY ?ind ?Face ?hasFrameNumber**
Query 2: Person – Running/Throwing/Kicking
SELECT ?Running ?ind ?hasFrameNumber **WHERE {** **?ind rdf:type SecurityDomain:Person .** **?ind SecurityDomain:hasAction ?Running .** **?ind SecurityDomain:hasFrameNumber ?hasFrameNumber .** **}** **ORDER BY ?Running ?ind ?hasFrameNumber**

Query 3: Person - Attacking

```
SELECT ?ind ?Person
  WHERE {
  ?ind rdf:type SecurityDomain:Attacking .
  ?ind SecurityDomain:isPerformedBy ?Person .
  }
ORDER BY ?Person
```

Query 4: Person – Kicking – Car/Person - Vandalism

```
SELECT ?ind ?Person
  WHERE {
  ?ind rdf:type SecurityDomain:Person .
  ?ind SecurityDomain:isKicking ?Vehicle
  }
ORDER BY ?ind
```

Query 5: Car – Fire/EventType - CriminalDamage

```
SELECT ?ind ?Fire ?hasFrameNumber
  WHERE {
  ?ind rdf:type SecurityDomain:Car .
  ?ind SecurityDomain:isOn ?Fire .
  ?ind SecurityDomain:hasFrameNumber ?hasFrameNumber
  }
ORDER BY ?hasFrameNumber
```

5.7. Visual Semantic Retrieval Results

5.7.1 Query Results for Running

As previously mentioned, 36 instances of *running* performed by different person in the video footage were annotated. The instances were labelled as R*unning_1, Running_2, Running_3* to *Running_36*, where each label represents a complete sequence of running action in consecutive frames performed by one person. Note that at this point, the person who executed these actions is unknown. Pallet reasoner was invoked to perform the reasoning process, make an inference and assert new knowledge in the ontology, based on the rules that have been created. Queries were then performed to extract inferred facts from the knowledge base. Based on queries of *Running_1* action to the inferred OWL ontology, 12 persons were detected from 22 frames. Extracted images were observed and analysed.

From the analysis, a sequence of ‘a running person’ is obtained from the identified frames. Based on the observation, one individual was repeatedly appeared in the collection of extracted images and was identified to be performing *Running_1.* Precision, recall and F1 score for *Running_1* detection is recorded to be 0.833 for all. Higher precision was produced due to the implementation of elaborate rules which performs detail features comparison in every frame, producing a higher accuracy of person detection and thus, low false positive. Higher recall showed that relevant detection has been retrieved in this experiment and F1 score indicated balance in the results distribution. The same process was executed to identify the individual who performed other actions. Figure 5.15 and 5.16 shows the selected frame sequence of action *Running_1* and *Running_27*, showing the person who performs it.

Figure 5.15: Sequence of Running_1 based on query results.

Figure 5.16: Sequence of Running_27 based on query results.

5.7.2 Query Result for Attacking

Attacking is defined as a person who performs *running* and *throwing* actions. *Attacking* action was inferred from asserted input through the rule-based inference process. The SPARQL query returned 667 inferred results related to *attacking* activity representing person and faces involved during the riot event. The data were analysed, and 12 occasions of the attack are identified in the

video footage. Based on *attacking* analysis, precision was recorded to be 0.153, recall is 0.867 and F1 score is 0.260. These results were produced as a consequence of an implementation of a more generic rule to classify an *attack.* The outcome of the reasoning process generated a higher volume of a detected *attack,* but with greater false positive results. This led to the low precision and F1 score. However, less volume of false negative generated higher recall, indicating relevant detection results of an *attack* action has been achieved. Four sequences of attack performed by different person are presented in Figure 5.17.

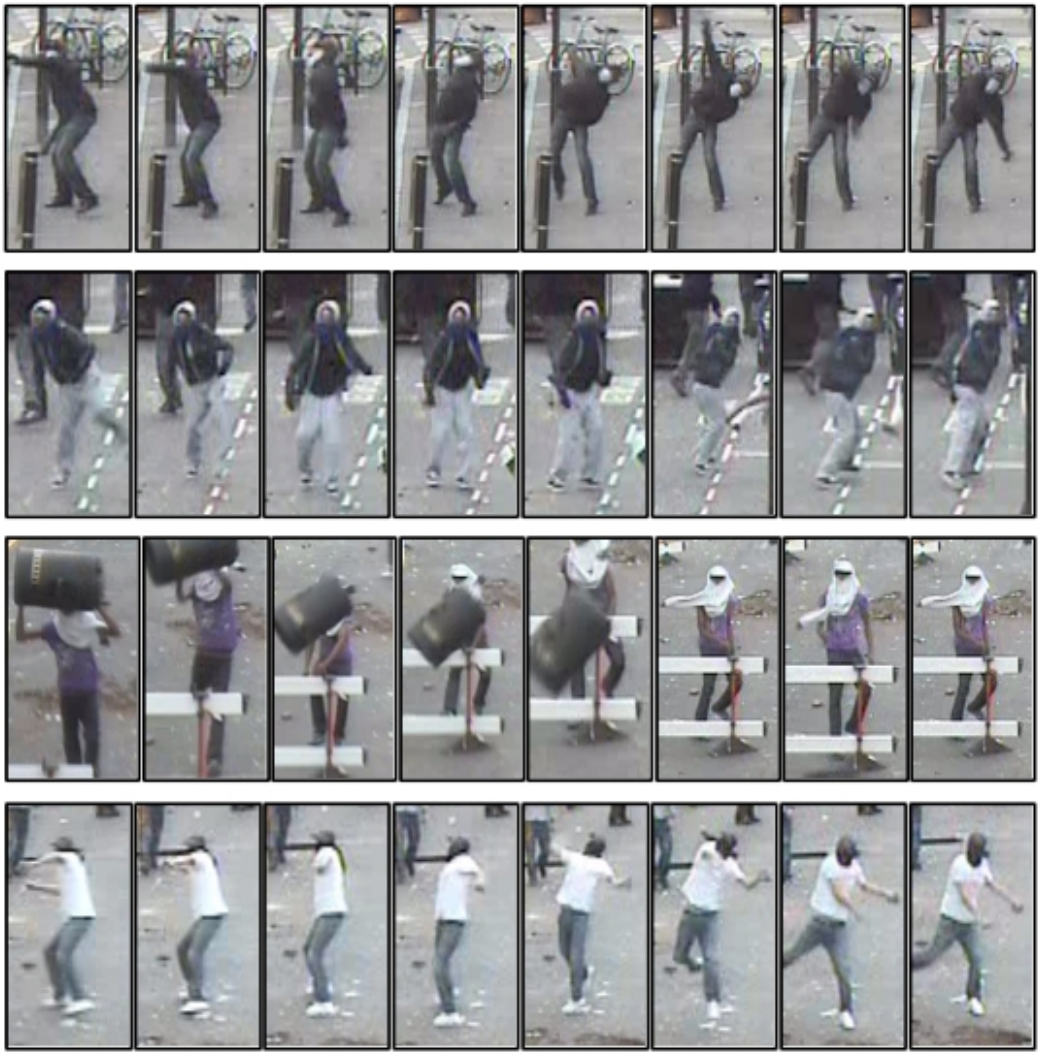


Figure 5.17: Attack sequences obtained from rule-based inference process

5.7.3 Query Result for “PersonisKickingACar’

There were three instances of *kicking* and 45 instances of car being annotated from the video footage. To derive the knowledge about the person who performs *kicking* and which car has been kicked, a rule is created to link the kicking action with the person who appeared during the action execution and the car that was present during that time. In this experiment, the results of precision is 0.118, recall is 0.923 and F1 score is 0.209. Lower precision and F1 score was resulted from an implementation of a rule to link three concepts (action, person and object) during the classification. This yielded higher false positive in the results. An elaborated rule is needed to produce a better detection result. Based on the analysis, the query results show that *Car_11* has been kicked by a man who was wearing a blue and white stripe hoodie as shown in Figure 5.18. A snippet of the query result is also shown in the figure. Note that the car has been cropped from the image by the person detector algorithm.

?ind	?Vehicle
untitled-ontology-14:Person-15014.png	untitled-ontology-14:Car_11
untitled-ontology-14:Person-15015.png	untitled-ontology-14:Car_11
untitled-ontology-14:Person-15016.png	untitled-ontology-14:Car_11
untitled-ontology-14:Person-15017.png	untitled-ontology-14:Car_11
untitled-ontology-14:Person-15018.png	untitled-ontology-14:Car_11

Figure 5.18: Kicking action extracted using queries to the knowledge base

5.8. Discussion

The results of the experimentation stage demonstrate that ontology-based visual analysis offers a promising result in analysing long content surveillance videos and at the same time extracting important high-level event from the video footage. This framework enables prominent features in a CCTV video to be represented in an ontology, while the process of understanding event semantics is achieved through the implementation of rule-based reasoning and queries. The results also shows extracted image of successful action and activities detection and its corresponding performance evaluation results. Based on the results, higher precision was achieved if a more elaborated rule was implemented in the system. Therefore, careful attention should be taken to design a more deliberate rule. By representing information from the low-level visual analysis in ontologies and incorporating high-level reasoning, rules and queries, benefits of using ontology-based knowledge representation and reasoning approach for visual analysis are tremendous.

5.9. Summary

This chapter demonstrated a visual semantic analysis, which implements information extraction from video footage of 2011 London riot and represent the knowledge in security domain ontology. A set of rules is proposed in this study to perform rule-based classification for knowledge inference. The experimental results show a successful action and activities detection, which accelerate the process of high level event detection and understanding from the video footage. The next chapter will address social media analysis to substantiate the importance of social media contents to facilitate event understanding.

CHAPTER 6

SOCIAL MEDIA ANALYSIS

This chapter presents experimental results for social media semantic analysis. The framework involves text annotation on a social media corpus using a language processing tool, parsing process for inter-level data transformation, ontology population, rule formation, and reasoning and query implementation for knowledge retrieval as shown in Figure 6.1. Related work on social media analysis is presented in Section 2.1 and language processing using GATE is summarised in Section 2.4.5. Text annotation process pipeline is elaborated in Section 3.4.2. In this chapter, social media analysis further extends visual semantic analysis by providing supplementary information about the event through social media users.

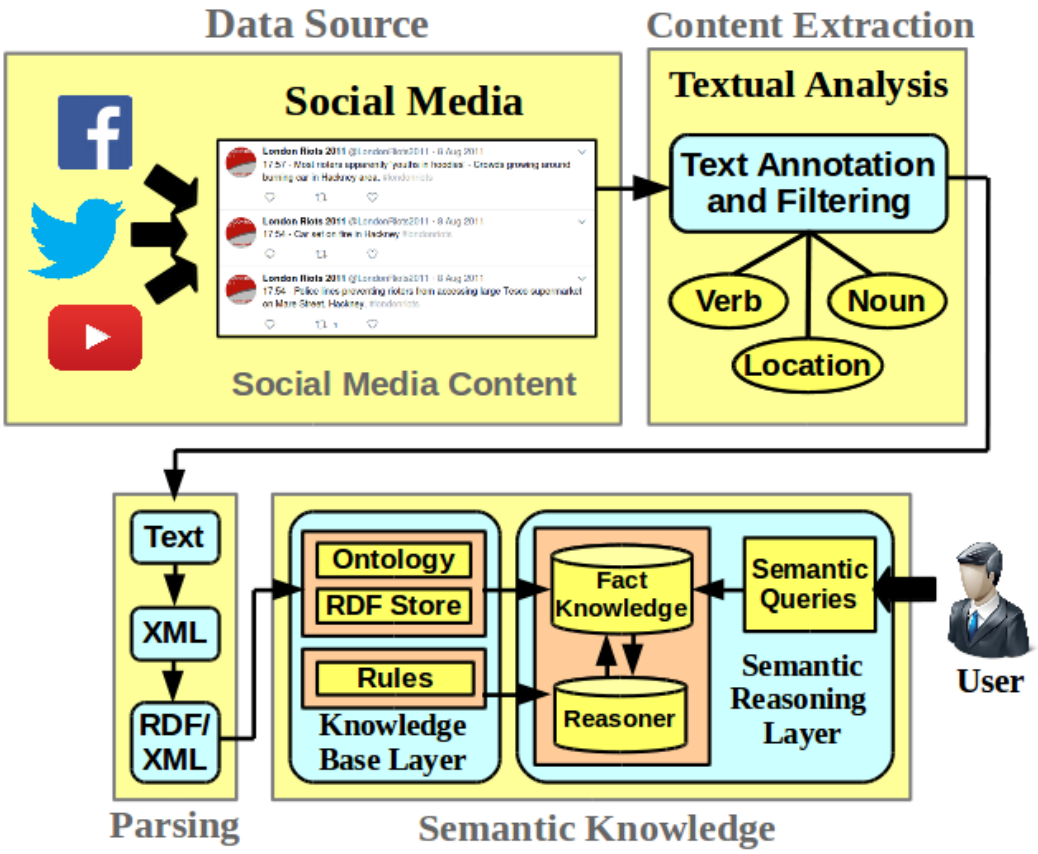


Figure 6.1: Social media semantic analysis framework

6.1. Social Media for Event Reporting

The substantial volume of useful information produced by the public's collective intelligence is highly beneficial for surveillance and investigative purpose. This effective and efficient event monitoring is made possible through extensive reporting by an active and ubiquitous community [22] of social media users, who act as a 'human sensor'. Human sensing produces a multi-perspective, multimodal user-generated content (UGC) in the form of descriptive text posts and event-oriented digital photos or videos, which could never be attained using a conventional sensor such as CCTV camera. This statement is substantiated by an example in Figure 6.2 showing four sample frames that are taken from CCTV footage of London riots event and tweets posted by social media users.

Figure 6.2: Photos of 2011 London Riots and social media shared contents produced by social media users

Based on this figure, it is apparent that social media user-generated content produced by the 'human sensor' presents more focused and comprehensive event annotations instead of a mere videos or photos. Nearly-real-time reports from human sensors' about on-the-ground situations such as locations, times and incidents have immense value for security forces and emergency authorities to assess events. By utilizing this information, emergency authorities will better understand 'the big picture' during critical situations, and thus make the best, most informed decisions possible for deploying aid, rescue and recovery operations [20].

6.2. Dataset

To perform social semantic analysis, a collection of tweets from twitter channel @LondonRiots2011, @londonriot, @london_riots11, @TheLondonRiots, @LDNRiots2011 and @LondonRiotsInfo that was actively reporting during the London riots event in August 2011 was compiled to create a Twitter corpus. The tweets include reports on the current situation, updates on travel disruption, safety reminders, public views on the incidents as well as the sharing of several speculated incidents during the event. Data from Twitter is specifically analysed for its large volume of data representing this event, where approximately 3.4 million people from the UK visited Twitter's homepage during the first day of the turmoil [127].

6.3. Ontology for Social Media Analysis

The social media analysis methodology consists of ontology engineering, data mining, concept mapping and knowledge inference and retrieval to analyse a large amount of social media data. The use of ontology has been proposed to bridge the gap between syntactic information retrieved from the data mining process and semantic concepts in the ontology. The security domain ontology has been developed based on the DOLCE foundational ontology [121] as it promotes flexibility for heterogeneous ontologies to inter-operate. This offers great benefit for the integration of NLP concepts extracted from social media data to be populated into existing security domain ontology. To perform social semantic analysis, the social media conceptual model for security domain ontology is defined. This model aims at addressing several elementary aspects of the event description for social media data and establishes a common foundation in the security domain. For this purpose, information included in user tweets during the 2011 London riots was analysed. Figure 6.3 depicts several Twitter post examples selected from the Twitter corpus.

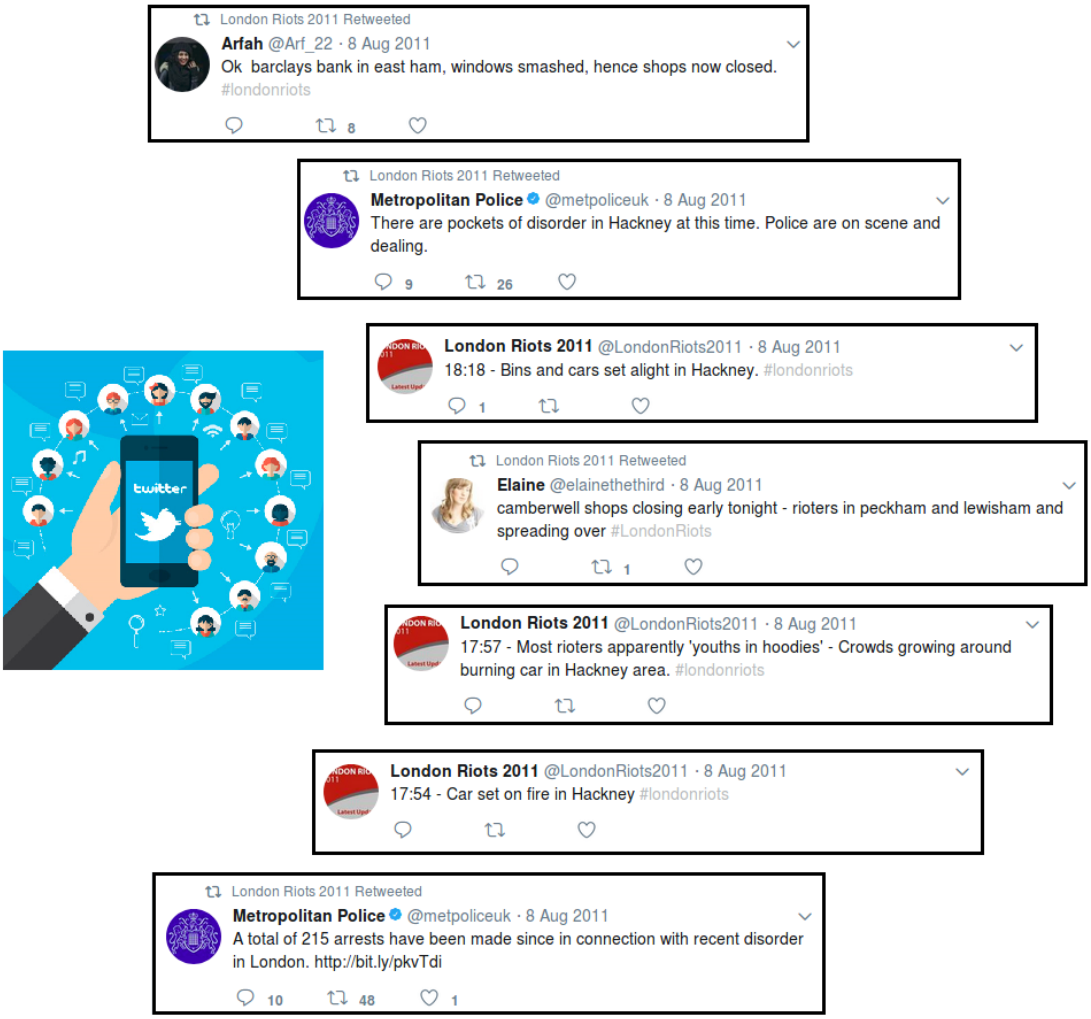


Figure 6.3: Example of Tweeter posts during 2011 London riots

For instance, the tweet “Car set on fire in Hackney” (two from bottom), provides information of location: Hackney; time: 17:54; noun: Car, fire; and verb: set; which contributes to a knowledge that there is an incident of a burning car in Hackney at 17:54. The Twitter post example also shows that major incident like fire is mentioned or tweeted multiple times by different users, which indicate the severity of the event and urgency of actions to be taken. Tweets from Metropolitan Police are an important and reliable source of information, which reaffirm reports that have been shared by the public at the event.

In a nutshell, these examples depict that user tweets carry valuable information about situations and location as well as time reference that can be used for event detection and understanding. Hence, semantic concepts such as Location and Time are defined in the ontology to describe the spatial and temporal aspects in the domain, and semantic concepts Verb [128] and Noun are defined to present informational aspects of event description. Concepts such as *Token, Sentence, StartNode, EndNode,* and *StartEndNode* are also specified to support social media analysis based on NLP, as presented in Section 4.1.3.2.

6.4. Text Annotation using GATE

As presented in Section 3.4.2, text annotation was executed using GATE software. In this study, GATE is implemented to annotate a Twitter corpus about the London riots event (Section 6.2). The annotation process using GATE software is illustrated in Figure 6.4. Each article in the corpus is linguistically pre-processed by performing tokenization, gazetteer, sentence splitting, POS tagging, NE transducing, Ortho matching, Tweet normaliser, Hashtag tokenization, Language Identification and Emoticons gazetteer to produced annotation sets.

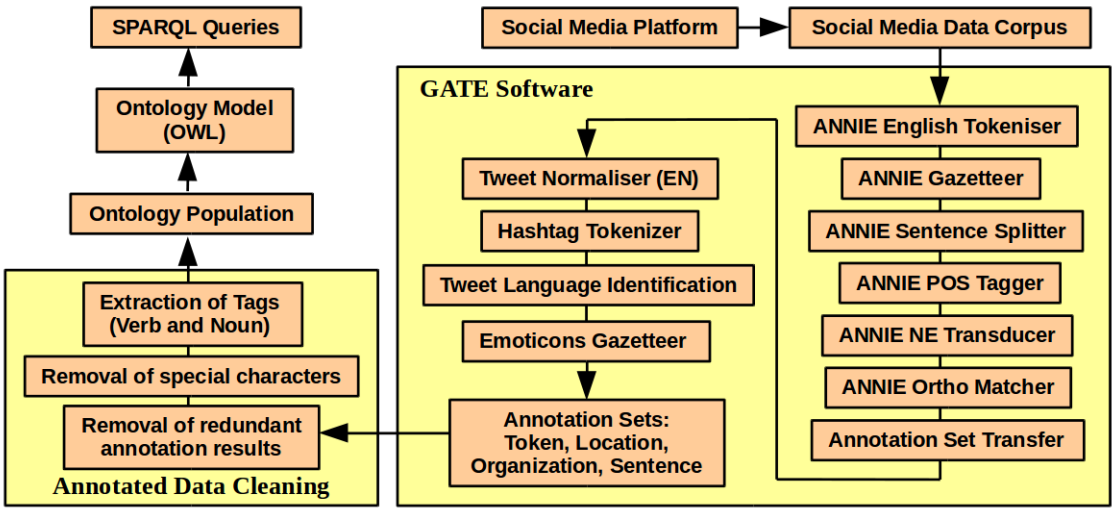


Figure 6.4: Textual data annotation and ontology population pipeline

Data cleaning process follows to remove unwanted annotation results. This includes redundant annotation results and special characters which were selected and removed manually by using Microsoft excel worksheet during the data cleaning process. Extraction of verb and noun from the resulted POS tags is also being done manually using Microsoft excel. The final results are then populated into the ontology for reasoning and query process.

1875 tweets from 8th August 2011 to 11th August 2011 was analysed. The annotation process produced 33,824 annotated tokens which include 27,999 words, 2631 numbers and 5338 symbols and punctuations. 11,498 nouns, 5769 verbs and 10,732 annotations for other POS tags were obtained for word annotation. 1400 locations, 682 organizations and 3092 sentences were also annotated. These annotated data was populated into the existing security domain ontology after post-annotation data cleaning has been performed. The newly populated ontology consists of 204,838 axioms, 14 classes, 11 object properties, 2 data properties and 44,819 individuals. The bar chart in Figure 6.5 summarizes annotation categories that have been processed from the Twitter corpus.

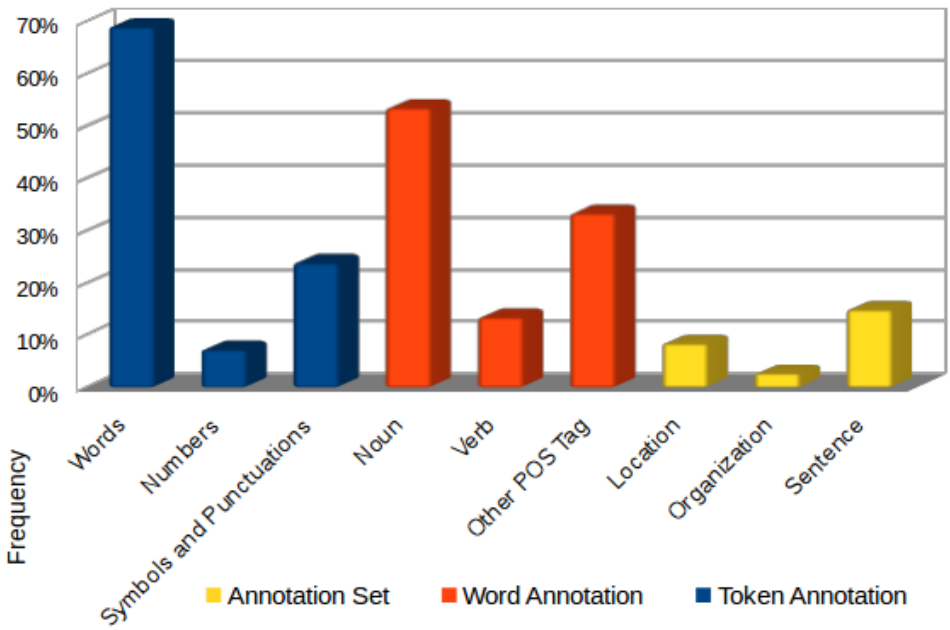


Figure 6.5: Annotation categories and tags from Twitter corpus

The annotated data from the Twitter corpus analysis was transformed into formal semantic data in the ontology through the parsing module. The parsing module undergoes several stages of transformation to parse a text document into an OWL ontology as described in Section 3.5. In the transformation process, GATE's annotation types were mapped to classes, annotated words were mapped to individuals and its characteristics were mapped to object properties and data properties. This newly generated Twitter corpus-based concepts and properties information were merged with the existing ontology to produce an ontology which represents both visual information and social media concepts and support semantic analysis in a comprehensive way.

6.5. Ontology Population

The annotated data obtained using GATE's ANNIE and Twitter plugins were mapped in the ontology as individuals of the similarly named concept. These concepts were defined as T*oken, Location, Organization, Sentence, Verb,* and *Noun*. The *Token* was divided into word, number, symbol, punctuation, and space token. Each word was annotated using POS tagger which is classified into noun, verb and other POS tags. *Noun* consisted of *Location* and *Organization* among other words. *Sentence* represented the information of each sentence number together with the start node and end node values of every sentence. The *StartEndNode* class represented the start and end node information for every individual retrieved from the source document.

By harvesting annotations information, the relationship between every token and its related description can be established. Data properties were defined as *hasStartNode* and *hasEndNode*. These properties represented a start node and end node value separately, and it is unique for every individual in the ontology. Object properties such as *hasLocationType*, *hasLocationName, hasNoun*, *hasVerb*, *hasStartEndNode*, *hasOrganizationType*, *hasOrganizationName* and *hasSentenceNumber* were used to link two related individuals together.

Figure 6.6 shows an example of a sentence in the Twitter corpus (top) and extracted details of its annotation sets (bottom table). Based on the annotation sets, the '*sentence'* starts at node 881 and ends at node 985 in the corpus. The '*token'* starts at node 972 to node 979 and was categorised as a '*Noun'* and has a string *'Hackney'.* The '*location'* also starts at node 972 to node 979 and was annotated with location type '*City'.* From this example, the token was assigned with *hasStartEndNode '972-979'.* Similarly, the location was also assigned with *hasStartEndNode '972-979'.* These properties were used to link instances by using semantic rules. Therefore, new knowledge was inferred which gave the location *'Hackney'* an object property *hasLocationType 'City'* and the noun *'City' hasLocationName 'Hackney'*.

17:57-Most rioters apparently 'youths in hoodies'- Crowds growing around burning car in Hackney area.

Type	Start	End	Details
Sentence	881	985	{ }
Token	972	979	{category=NNP, kind=word, length=7, orth=upperInitial, string=Hackney}
Location	972	979	{kind=locName, locType=city, matches=[25186, 25199, …, 25660], rule=InLoc1, ruleFinal=LocFinal}

Figure 6.6: Extracted annotation sets from a sentence in the Twitter corpus

6.6. Rule-based Inference and Queries

The rule-based inference is used to link information and assert new knowledge in the ontology. The object property *hasStartEndNode* and data properties *hasStartNode* and *hasEndNode* are exploited in SWRL rules to link instances, perform categorization and draw conclusions from the inferred ontology. The SPARQL is used to perform queries and retrieve information from the inferred knowledge base. The integration of rules and queries enables various event information such as location type, location name and authorities' involvement

during the event to be inferred and retrieved. A specific token relation such as the link between the noun *'fire'* and location *'Hackney*' and other corresponding concepts and instances relation are also being established using rules and queries to provide further information about the event.

SWRL rules were created for instances categorization and knowledge assertion in the ontology. As discussed in Section 6.5, every instance that represents Token, Location, Organization, Verb, and Noun that was annotated using GATE, were assigned with *hasStartNode* and *hasEndNode* data properties and *hasStartEndNode* object property. These attributes were used to link instances and establish relationships between them. By referring to Figure 6.6 once again, an annotated '*location'* of location type *'City'* carried attribute *hasStartEndNode '972-979'* and other annotated '*token'* of string 'Hackney*'* carried the same attribute *hasStartEndNode '972-979'*. Using inference rule, both instances were matched and asserted with new object properties, thus creating new inferred knowledge *'Hackney' hasLocationType 'City'* and *'City' hasLocationName 'Hackney'*. This explains Rule 1 and Rule 2 as follows:

Rule1:
Token(?A),hasStartEndNode(?A,?X),Location(?B), hasStartEndNode(?B,?X)->hasLocationType(?A,?B), hasLocationName(?B,?A)
Rule2:
Token(?A),hasStartEndNode(?A,?X),Organization(?C), hasStartEndNode(?C,?X)->hasOrganizationType(?A,?C), hasOrganizationName(?B,?A)

Rule 3 was created to identify all sentence numbers which contains the token *'fire'.* The token *'fire'* was chosen because it represents one of the highlights during the riot event. Rule 4 and 5 were used to distinguish *verbs* and *nouns*

that co-existed with the word '*fire'* in a sentence to attain additional information that describes the situation. For instance, *'What object is on fire?'* and *'Where does the fire incident happen?'*. Rule 3 to Rule 5 which were used to infer new instance properties related to the token '*fire'* are as follows:

Rule3:
Token(fire),hasStartNode(fire,?W),hasEndNode(fire,?X), Sentence(?D)hasStartNode(?D,?Y),hasEndNode(?D,?Z), swrlb:greaterThanOrEqual(?W,?Y),swrlb:lessThanOrEqual (?W,?Z),swrlb:greaterThanOrEqual(?X,?Y), swrlb:lessThanOrEqual(?X,?Z) -> hasSentenceNumber(fire,?D)
Rule4:
Token(fire),hasSentenceNumber(fire,?D),Sentence(?D), hasStartNode(?D,?Y),hasEndNode(?D,?Z),Verb(?E), hasStartNode(?E,?U),hasEndNode(?E,?V), swrlb:greaterThanOrEqual(?U,?Y),swrlb:lessThanOrEqual (?U,?Z),swrlb:greaterThanOrEqual(?V,?Y), swrlb:lessThanOrEqual(?V,?Z) -> hasVerb(?D,?E)
Rule5:
Token(fire),hasSentenceNumber(fire,?D),Sentence(?D), hasStartNode(?D,?Y),hasEndNode(?D,?Z),Noun(?F), hasStartNode(?F,?P),hasEndNode(?F,?Q), swrlb:greaterThanOrEqual(?P,?Y),swrlb:lessThanOrEqual (?P,?Z),swrlb:greaterThanOrEqual(?Q,?Y), swrlb:lessThanOrEqual(?Q,?Z) -> hasNoun(?D,?F)

The process started by identifying all sentence numbers which contains the token *'fire'.* This was done by comparing the token's and sentence's *StartNode* and *EndNode.* If the token's *StartNode* and *EndNode* lie in between the sentence's *StartNode* and *EndNode,* the sentence was classified as having a token *'fire'.* After Rule 3 was executed, all sentence numbers which contains the token *'fire'* were identified. Then, Rule 4 and Rule 5 helps to find all verbs and nouns that were mentioned in those sentences. Similarly, the process was done by comparing the verb's and noun's *StartNode* and *EndNode* with the

sentence's *StartNode* and *EndNode.* Enquiries for *'Sentence', 'Verb'* and *'Noun'* in SPARQL query helps to retrieve all verbs and nouns that exist in the same sentence as the token *'fire'* in order to further understand situations during the fire event. Table 6.1 shows an example of the SPARQL query to retrieve inferred knowledge from the ontology.

Table 6.1: SPARQL query for concept retrieval

SPARQL Query:

```
PREFIX owl:<http://www.w3.org/2002/07/owl#>
PREFIX rdf:<http://www.w3.org/1999/02/22-rdf-syntax-ns#>
PREFIX rdfs:<http://www.w3.org/2000/01/rdf-schema#>
PREFIX xsd:<http://www.w3.org/2001/XMLSchema#>
PREFIX SecurityDomain: <http://www.semanticweb.org/farhan/
ontologies/SecurityDomain#>

SELECT  ?Verb ?Noun ?ind
   WHERE {
      ?ind rdf:type SecurityDomain:Sentence .
      ?ind SecurityDomain:hasVerb ?Verb .
      ?ind SecurityDomain:hasNoun ?Noun .
      }
   ORDER BY ?Verb ?Noun ?ind
```

6.7. Social Semantic Analysis Results

In accordance with rule-based inference and semantic query discussed in Section 6.6, social semantic analysis results were presented. Several keywords were chosen in relation to the event that has been presented in the visual semantic analysis to examine the correlation between visual and textual analysis results and to draw supplementary information regarding the event. The chosen keywords describe actions *Running, Throwing,* and *Kicking,* activity *Attack, Smash* and *Loot* and keyword describing *Fire* event. Two highly

mentioned locations *Hackney* and *Croydon* were also chosen to investigate related verb and noun associated with those locations. Extracted keywords were classified as Action, Object, People, Place, and Location category.

6.7.1 Query results for Running, Throwing and Kicking actions

Three actions *Running, Throwing* and *Kicking* that were performed during the riot was chosen to be queried and analysed. Based on the results in Table 6.2, *running* action has been carried out on the road and streets in *Hackney, Holburn, Einfield,* and *Birmingham*. In addition to getting information about people involved in this action, the results also displayed some objects that were carried by the runner which are *wood.* Furthermore, the query results for the keyword *throwing* exemplified the kind of objects used to attack the police. The objects include *bottles, glass, bricks, missiles,* and *stone.* It is also shown that people have targeted *vans, cars* and *vehicle* to be thrown at and this action reportedly happened in *Hackney* and *Glasgow*. Finally, based on *kicking* results, no significant information has been shared apart from the location where it happened.

Table 6.2: Query results for running, throwing and kicking actions

Action	Object	People	Place	Location
running	- wood	- People	- road - streets	- Hackney
running	- fire	- guys	- streets	- Holburn - Einfield
(being) run		- Muslim men - rioters		- Birmingham
throwing	- bottles - glass	- police		- Glasgow
(being) thrown	- bricks - missiles - vans	- police		- Hackney
throwing	- cars - stones - vehicles	- police		

kicking	- news	- BBC	- Leeds - Peckham - Slough
kicked			- Birmingham - Leeds
kick	- water cannons	- Home Secretary	

6.7.2 Query results for Attack, Smash and Loot activities

Table 6.3 presents the place, people and object involved in an attack. It can be seen that *hospital, police station, premises* and *shops* were among the targeted place and *bricks* were used during an attack. Based on the noun-verb analysis, attack activity were also associated with *riots, violence, fire* and *looting.* Additionally, for *smash* activity, the keyword *windows* were repeatedly encountered during the queries. Therefore, it can be certained that *windows* were being smashed during the riot. Places involved includes *BP, Children's Hospital, Barclays bank, shops* and *JD store.* Further analysis needs to be carried out to identify in which place the windows were smashed by the youth rioters.

Table 6.3: Query results for attack, smash and loot activities

Activity	Object	People	Place	Location
attack (riots)	- bikes - head - video	- cops - kids	- hospital - police station	- Manchester - Croydon - Springbridge Road - Nottingham Canning Circus
(been) attacked (violence)		- police	- capital city - premises - shops	- Birmingham
(being) attacked				- Ealing - Thronton Heath
attacked (fire)	- bricks	- police - witnesses		- Gloucester

attacked (looting)	- bricks - fire - truck	- Bruneian - people - witnesses	- streets	- Croydon - Gloucester - London
attacking		- rioters - civilians	- street - businesses	- Ealing - London
smashed	- windows	- youths - police - rioters	- BP - Children's Hospital - Barclays bank - shops - JD - store	- Eastham - Edmonton - Birmingham - West Bromwich - Camden - Manchester
(being) smashed	- windows	- youths	- BP	- Edmonton - West Bromwich
(getting) smashed			- shops	- Eastham
smash	- window		- Ladbrokes	- Hackney
smashing	- vans	- police - rioters		- Bristol

Looting activity is a complex high-level event and difficult to be inferred semantically in visual semantic analysis. However, using social media posts provided by social media users, information about looting activities can be acquired effectively. Based on Table 6.4, people were seen carrying *alcohol, cigarettes* and *other things* out from the shops in *Birmingham, Barking* and *Romford.* Looters were also occupied by *Molotov cocktails, poles* and battering rams. The keyword *looters* also associated to other activities such as *broken premises, smashed shops* and *stealing from a man.*

Table 6.4: Query results for looting activity

Activity	Action	Object	People	Place	Location
looted	carrying: - alcohol - cigarettes - other things	- bags	- teens - youths	- Santander bank - Asda - JD - Primark - pubs	- Birmingham - Barking - Romford

			- shops - mobile phone - shops	
looters	using: - Molotov cocktails - poles - battering rams	- car	- Tesco	- Uxbridge
	broke:		- premises	- Huddersfield - Manchester - Salford
	smashed:		- shops	- Camden - Eastham
	stealing:	- man		- Hackney
looting (burning/ fire)		- cars	- Co-op - shops	- Liverpool - Lawrence Rd. - London
looting		- cars	- shops	- Birmingham - Bullring Centre - Liverpool - London - Smithdown Rd. - Woolwich - Brixton - Tottenham

6.7.3 Query results for Fire event

Based on the analysis, the keyword *fire* is associated with other verbs and nouns such as *burning, confirmed, blaze, tackling, spread, fighting,* and *set.* Objects that were on fire are identified as *car, bin, vehicles, building, bus* and *windows* and several places have been named related to *fire* event such as *Sony Distribution Centre, Tesco, Greg's Baker* and etc. According to Table 6.5, more than 30 locations have been named, relevant to a fire event.

Table 6.5: Query results for the token Fire

Fire	Object	People	Place	Location
fire (burning)			- Sony Distribution Centre	- Enfield - North London - Croydon
fire (confirmed)	- car - bin		- car park - Tesco	- Croydon - Elmers End
fire (blaze) (tackling)			- Town Centre	- London - Peckham
fire (spread)	- vehicles	- police	- city	- Liverpool
fire (fighting)	- pockets of fire	- fire crews	- shops	- Croydon
fire (set)	- buildings - bus - car - shop - vehicles	- arsonist - police	- Co-op - Westfield	- Brixton - Croydon - Edmonton - Hackney - Lawrence Rd. - Lewisham - Liverpool - London - Wood Green - Claphamjunction
fire	- buildings - bus - car	- police - society - fire crews	- Bookmakers - Sony Distribution - Greg's Baker - shop - Tesco - supermarket - residential - premises	- Bethnal Green - Bukinghamshire - Birmingham - Ealing - Enfield - Kent - Peckham - Waltham Abbey - Walthamstow - Lavender Hill - Woodford - Woolwich
alight (set)	- bins - buildings - bus - cars - vehicles - windows	- community - youths	- city centre - London Eye - Miss Selfridge - shop - store - street	- Croydon - Dartford - Hackney - London - Manchester

6.7.4 Query results for location Hackney and Croydon

Analysis of the corpus was also carried out to determine the location which was frequently mentioned on Twitter within the time frame of the event. Based on the query results, London was mentioned most frequently in the corpus followed by Hackney, Croydon, Peckham, Lewisham and Birmingham. It is interesting to note that Birmingham was mentioned a few times in the Twitter messages despite being located far from London. This indicates that a riot happened in Birmingham at the same time the Twitter post was made. By acquiring these information, security forces can identify locations which is severely affected by the riot and act accordingly. Table 6.6 presents event description based on queries of Hackney and Croydon and Figure 6.7 depicts four locations which were severely affected based on query results and the Twitter post related to each location.

Table 6.6: Event description based on queries of Hackney and Croydon

Location	Action	Object	People	Place	Location
Hackney (burning) (growing)		- car	- crowds - rioters - hoodies - youths		
Hackney (fire)		- car - fire			
Hackney (disrupted)				- London Overground services	- Barking - Hackney Central - West Croydon
Hackney (police)	- firing - patrol	- fireworks - rioters - youths	- police officers - Welsh police	- Overground station - Underground station	- Brixton
Hackney (injured) (stealing)		- CCTV footage	- looters - mans		

Hackney	-hurl	- anything	- police - rioters	
	- running	- wood	- people	- streets - road
	-smash	- window	- people	- Ladbrokes
	- thrown	- missiles		

Location	Object	People	Place	Location
Croydon (fire)	- buildings - car	- arsonists - fire crews - rioters	- Bookamakers - furniture store - House of Reeves - premises - residential	- South London
Croydon (burning)	- severe fires			
Croydon (confirmed)	- car - fire			
Croydon (evacuates)	- homes		- residents	- Clapham
Croydon (fighting)	- fire	- fire crews	- flats - shops	
Croydon (arrest)	- fire		- furniture store - House of Reeves	- South London
Croydon (shooting) (died)	- car	- man - Met Police	- hospital	
Croydon (police)		- police	- home - Underground station	- Barking - West Croydon - London

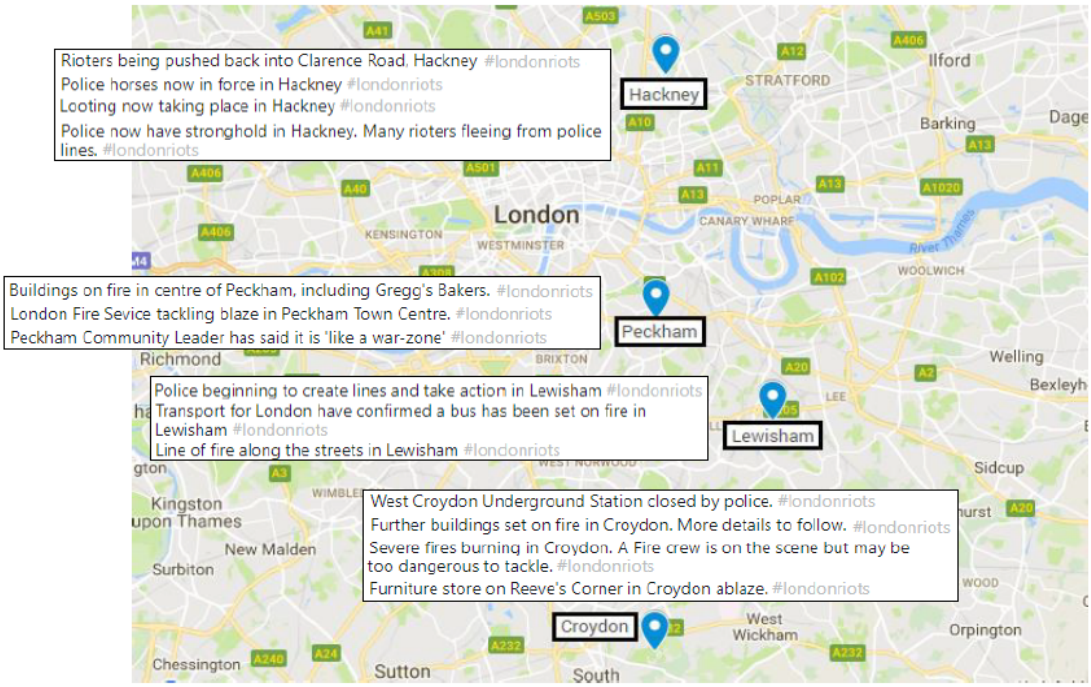


Figure 6.7: Locations severely affected by the riot based on query result

Spider diagrams in Figure 6.8 shows a mapping of concept-noun relations retrieved from queries on the knowledge base. The words ‘fire’, ‘Hackney’, ‘smashed’ and ‘looted’ are the keywords selected in this example because it is frequently mentioned in the tweets and carries a significant meaning in the event. The red labelled noun indicates the important word, such as location; a repeated word; or it represents an important description of the event. The blue labelled noun is the similar word that was retrieved from queries of multiple keywords. The ones labelled in green, although it is less significant, gave additional information related to the queried keyword. Figure 6.8 (top-left) described the related nouns for keyword ‘fire’. The result suggested a few locations where a fire has emerged, which was ‘Hackney’, ‘Peckham’, ‘Croydon’, ‘Lewisham’, ‘Enfield’ or/and ‘London’. It also indicated that ‘vehicle’, ‘bus’ or ‘building’ was on fire as these nouns occurred repeatedly in the result.

Query results for ‘Hackney’ in Figure 6.8 (top-right) support the previous claim that fire has emerged in Hackney where the noun ‘fire’ and ‘burning’ were

retrieved, together with other nouns such as ‘shop’, ‘Tesco’, ‘Station’ and ‘car’. The word ‘car’ occurs three times when ‘Hackney’ is queried. This indicates that a car was on fire in Hackney instead of a bus which was initially presumed. Other important descriptions about the severity of the event in Hackney can be seen from the retrieved nouns ‘police’, ‘rioters’, ‘disorder’, ‘disturbances’, ‘looting’/’looters’ and ‘alight’. Figure 6.8 (bottom) illustrates retrieved nouns resulted from queries of keywords ‘smashed’ and ‘looted’. The result shows that ‘shops’, ‘bank’ ‘City’, ‘youths’ and ‘colleague’ were common retrieved nouns for both keywords. This indicates that ‘shops’ and ‘bank’ were among premises that have been ‘smashed’ and ‘looted’ by some ‘youths’. This result also indicates that the ‘window’ of ‘barclay’, a ‘Hospital’ or ‘BP’ has been smashed and ‘Santander’, ‘Asda’, ‘Ladbroke’ or some ‘pubs’ have been looted.

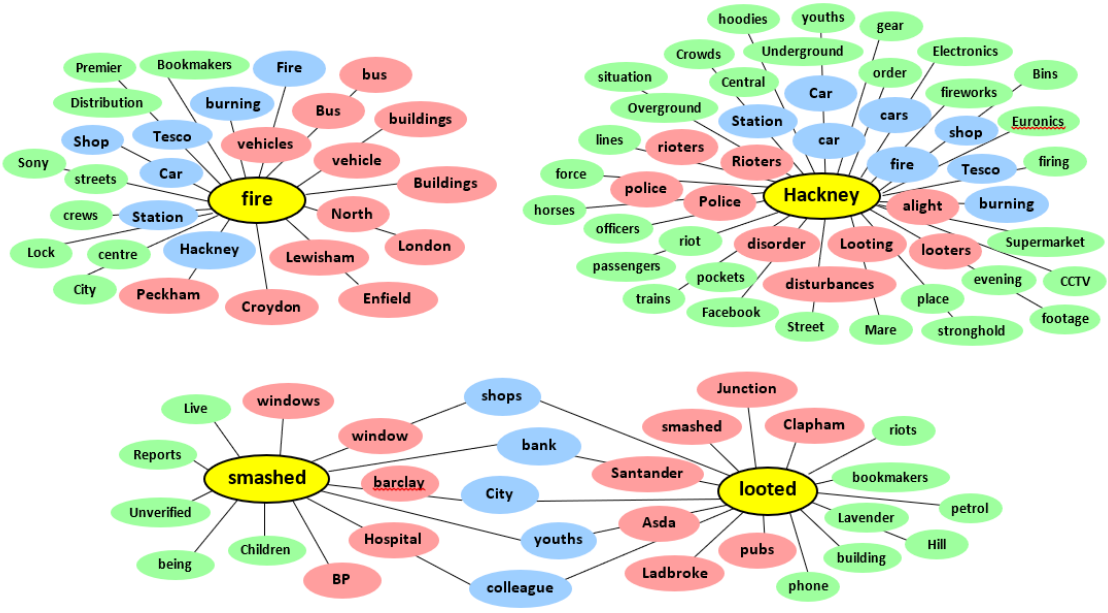


Figure 6.8: Keywords and retrieved nouns from the Twitter corpus

6.8. Discussion

The analysis that has been performed demonstrated that the information supplied by the human sensor through social media provides valuable information about on-the-ground situations to assist security forces to act

promptly during a crisis. The analysis yields promising results using only five GATE annotated concepts from the Twitter corpus of the riot event. However, there remain many hurdles for optimal exploitation of social media. Huge limitations of using data originated from social media includes unknown reliability and accuracy of information shared in social media and the usage of ill-formed text (lower-casing names, doubling the letter for stress and misspell words) in social media messages. These factors reduce the annotation recall and accuracy during annotation task, thus affects the reasoning process of data and overall detection results. Therefore, improvement should be made in a few aspects such as information filtering, relevant information ranking, and standard terminologies compliance in order to improve the accuracy of results, hence producing a better comprehension of the event.

CHAPTER 7

CONCLUSION

Recent developments in multimedia technologies have led to the generation of vast quantities of multimedia data from a variety of sources and sensors. However, when the research presented in this thesis began, there was still a significant gap in the automated understanding of very large amounts of data capturing single real-world events. Thus, in this thesis, a security domain ontology framework for event detection and understanding is presented to address and propose a solution to this problem. Various techniques to extract visual and textual semantic cues have been introduced and unified into one hybrid stream representing a comprehensive framework. It supports substantial data volume from media sources including CCTV and social media networks.

The proposed security domain surveillance system framework was introduced in Chapter 3 incorporating four main modules: the Data Source module, Content Extraction module, Parsing module and Semantic Knowledge module. The data source and content extraction module centralize on data acquisition and salient information extraction through visual analysis of video footage and textual analysis of social media content. The visual analysis module fulfilled the automated salient feature extractions from the video footage by implementing HOG and Haar-feature based cascade classifier approach. Manual video annotation is implemented using ViPER annotation tool. Features that were being annotated are Person, Face, Police, Car, Fire and Running, Kicking and Throwing actions. Text annotation on Twitter data corpus is being executed using GATE supported by the ANNIE plugin. The annotation process in GATE follows through a corpus pipeline of ANNIE resources. During annotation, each article in the corpus is linguistically pre-processed by performing fine-grained tokenization, gazetteer, sentence splitting, POS tagging, NE transducing, Ortho matching, Tweet normalising, Hashtag tokenization, Language Identification, and Emoticons gazetteer to produced annotation sets. Annotation sets include Location, Sentence, Token, Verb, Noun, and Organization.

Both visual and textual annotation process produce feature description in textual format. The parsing module handles inter-level data transformation, aiming to transform syntactic information obtained from the Content Extraction module to high-level semantic concept representation in the Semantic Knowledge module. This procedure transforms XML documents to an OWL ontology. Finally, the semantic knowledge module manages all the semantic operations by performing higher-level event representation, knowledge reasoning, and queries. The semantic knowledge module consists of a knowledge base layer and a semantic reasoning layer. In the knowledge base layer, domain ontology and RDF store are employed for high-level representations; and rules are created for rule-based classifications. SPARQL enables users to query information from the knowledge base or any data source

that can be mapped to RDF. The queries are used to retrieve the in-memory triples in the newly inferred knowledge.

Chapter 4 highlights the development of an event conceptual model for security domain ontology through an implementation of ontology development methodology and the conceptualization classification extension in accordance with six elementary aspects which underpin functional requirements of an event model. The elementary aspects of event description promote a model that supports a common foundation for a wide diversity of applications, reusability and application integration. The basic aspects incorporate temporal, spatial, informational, experiential, structural and causal aspect of the event description. The ontology development methodology proposed in METHONTOLOGY were adapted to serve as a guideline for developing the security domain ontology. By adapting a mature methodology, the quality of the implemented ontology will be enhanced.

Based upon conceptualization discussion in Section 4.1.1, the security domain ontology conceptual model was tailored in conformity with elementary aspects of event description, along with an adaptation of DOLCE foundational ontology in the process of domain modelling. The DOLCE foundational ontology is implemented to classify every concept behind the ontological modelling decisions. Foundational ontologies act as a reference that commits to certain theories and provides a set of formal guidelines for domain modelling, and serve as a tool for making heterogeneous ontologies interoperate or merge.

Chapter 5 presents experimental results for visual semantic analysis which involves every stage in the security domain surveillance system framework. This includes data acquisition from CCTV footage, features extraction, parsing process, semantic reasoning and queries for knowledge retrieval. The dataset for visual semantic analysis contains the video footage on 2011 London Riots

obtained from Scotland Yard. The footage was captured in various locations across London during the riots. The video dataset was carefully analysed and relevant videos were selected and used to validate the developed system framework. For visual semantic analysis, 10,855 frames of CCTV footage which represent various situations during the event were used. The analysis process started with extractions of features using various video processing techniques and followed by textual to OWL data parsing for ontological analysis. A set of rules was proposed to model all the situations in the scene according to the events represented by the ontology. SWRL rules were adopted to build reasoning rules in order to represent the dynamic aspect of the surveillance system. During reasoning, inferences were made, classifying the instances of the security domain ontology and associating new properties to instances.

Semantic queries were used to retrieve the in-memory triples in the newly inferred knowledge. A knowledge base query will include facts that are inferred as well as facts that have been explicitly asserted in the knowledge base. SPARQL provides a formal language to ask meaning-driven questions in the knowledge base and is used to express these queries. Query results for Running, Attacking and PersonisKickingACar shows that ontology-based surveillance system was able to analyse and extract high-level events represented in the ontology.

Based on the experiment to retrieve *Running_1* action in the video footage, the precision, recall and F1 score for the detection was 0.833 respectively. The result shows a good detection and relevant knowledge retrieval has been made using rules and semantic reasoner. However, low precision and F1 score results was obtained for *Attacking* and *PersonisKickingACar* detection. The results were 0.153 and 0.260 for *Attacking* detection and 0.118 and 0.209 for *PersonisKickingACar* detection. This is due to simpler rules that was used compared to a more elaborated one used to detect *Running_1*. However, higher

recall was obtained for *Attacking* detection (0.867) and *PersonisKickingACar* detection (0.923). This experiment shows that the rules design plays an important role in producing a better detection result.

Chapter 6 presents experimental results for social media semantic analysis. The process involves text annotation using social media corpus, parsing process, ontology population, rule formation and finally reasoning and query implementation for knowledge retrieval. To perform social semantic analysis, a collection of tweets from Twitter channel @LondonRiots2011, @londonriot, @london_riots11, @TheLondonRiots, @LDNRiots2011 and @LondonRiotsInfo that was actively reporting during the London riots event in August 2011 was compiled to create a Twitter corpus. The tweets include reports on the current situation, updates on travel disruption, safety reminders, public views on the incidents as well as the sharing of several speculated incidents during the event.

The twitter annotation process was performed using GATE software. Tweets from the first four days of the riot was analysed. The annotation process produced 33,824 annotated tokens which include 27,999 words, 2631 numbers and 5338 symbols and punctuations. 11,498 nouns, 5769 verbs and 10,732 annotations for other POS tags were obtained for word annotation. 1400 locations, 682 organizations and 3092 sentences were also annotated. These annotated data are populated into the existing security domain ontology after post-annotation data cleaning has been performed.

Analysis of the corpus was carried out to determine the location which was mentioned repeatedly on Twitter within the time frame of the data. Based on the query results, London has the highest mentioned location in the corpus, followed by Hackney, Croydon, Peckham, Lewisham and Birmingham. The analysis has also been conducted with several keywords to perform a concept-

noun relations mapping on the knowledge base. The words ‘fire’, ‘Hackney’, ‘smashed’ and ‘looted’ were among the selected keywords because it was frequently mentioned in the tweets and carries a significant meaning in the event.

RECOMMENDATIONS AND FUTURE WORK

From this point of view, further research could be conducted on the following aspects:

1. The presented research in this thesis focuses on CCTV video footage and textual data from social media user-generated content. The current framework could be extended to include other sources of multimedia data such as images from social media platforms, videos captured using handheld devices and audio recording taken by spectators during the event and etc.
2. The extension can also be applied to new application scenarios as well as other domains knowledge that can benefit from the structured knowledge representation and reasoning which involve heterogeneous data sources. A more recent and better performance object detection and action recognition techniques [129] could be adopted to explore visual cues from the video footage rather than performing manual feature annotations. This approach could produce a better and more realistic outcome and improve overall results to achieve better video understanding.
3. As an alternative to RDF triple-store, property graph [130] could be implemented for knowledge representation and reasoning process. Property graph are known for its rich internal data model structure and the differences in size, with RDFs being an order of magnitude bigger than a property graph in many cases. Although the potential is there, more studies are needed for further implementation.

4. Unknown reliability and accuracy of information shared on social media and the usage of ill-formed text in social media messages affects the reasoning process of data and overall detection results. Therefore, improvement such as information filtering, relevant information ranking, and standard terminologies compliance should be made in order to improve the annotation accuracy, hence producing a better comprehension of the event.
5. Ontology-based approach can be implemented for content indexing in diversity of application domains like sport, broadcasting, news, cooking, etc. It could serve as an effective indexing tool to improve indexing consistency in manual annotation systems and propagation of labels in automatic indexing systems.
6. Ontology-based surveillance system can also be applied in a medical surveillance domain where research on Biosurveillance Application Ontology [131] has been proposed as an effective syndromic surveillance system to identify and monitor disease outbreaks in their early stages.

Adopting ontology-based approach, this thesis proposed a novel framework for automated media analysis in a security domain. The framework enables a large amount of video and social media data to be analysed systematically and automatically, and promotes a better method for high-level event detection and understanding. Lying on the crossroads of Visual Analysis and NLP, the information from both data sources render ‘the big picture’ during critical situations and thus help security forces and emergency authorities make the best decisions possible for deploying aid, rescue and recovery operations during the event.

REFERENCES

[1] Z. Xu, Y. Liu, L. Mei, C. Hu, and L. Chen, “Semantic based representing and organizing surveillance big data using video structural description technology,” *J. Syst. Softw.*, vol. 102, pp. 217–225, 2015.

[2] British Security Industry Association, “The Picture is Not Clear: How many CCTV surveillance cameras in the UK,” 2013.

[3] “How Many CCTV Cameras in London?,” *Caught on Camera*. [Online]. Available: https://www.caughtoncamera.net/news/how-many-cctv-cameras-in-london/. [Accessed: 15-Jul-2018].

[4] J. Cropley, “Top Video Surveillance Trends For 2018,” *IHS Technology*, 2018.

[5] “Number of monthly active Twitter users worldwide from 1st quarter 2010 to 1st quarter 2018 (in millions),” *In Statista - The Statistics Portal*. [Online]. Available: https://www.statista.com/statistics/282087/number-of-monthly-active-twitter-users/. [Accessed: 15-Feb-2018].

[6] “Number of monthly active Facebook users worldwide as of 1st quarter 2018 (in millions),” *In Statista - The Statistics Portal*. [Online]. Available: https://www.statista.com/statistics/264810/number-of-monthly-active-facebook-users-worldwide/. [Accessed: 15-Feb-2018].

[7] F. Pai, L. Yang, and Y. Chung, “Multi-layer ontology based information fusion for situation awareness,” *Appl. Intell.*, vol. 46, no. 2, pp. 285–307, 2017.

[8] N. Jenkins, “Video Surveillance Camera Installed Base Report - Industrial, Security and Medical | Video Surveillance,” *IHS Technology*, 2015.

[9] “Rise of surveillance camera installed base slows,” *SDM Magazine*, 2016. [Online]. Available: https://www.sdmmag.com/articles/92407-rise-of-surveillance-camera-installed-base-slows.

[10] L. Calavia, C. Baladrón, J. M. Aguiar, B. Carro, and A. Sánchez-Esguevillas, “A semantic autonomous video surveillance system for dense camera networks in Smart Cities.,” *Sensors (Basel).*, vol. 12, no. 8, pp. 10407–10429, 2012.

[11] R. Martínez-Tomás, M. Rincón, M. Bachiller, and J. Mira, “On the correspondence between objects and events for the diagnosis of situations in visual surveillance tasks,” *Pattern Recognit. Lett.*, vol. 29, no.

8, pp. 1117–1135, 2008.

[12] X. Zhou and L. Chen, “Event detection over twitter social media streams,” *VLDB J.*, vol. 23, no. 3, pp. 381–400, 2014.

[13] M. Brenner and E. Izquierdo, “Social event detection and retrieval in collaborative photo collections,” in *Proceedings of the 2nd ACM International Conference on Multimedia Retrieval - ICMR ’12*, 2012.

[14] J. R. Cózar, N. Guil, J. M. González-Linares, E. L. Zapata, and E. Izquierdo, “Logotype detection to support semantic-based video annotation,” *Signal Process. Image Commun.*, vol. 22, no. 7–8, pp. 669–679, Aug. 2007.

[15] J. Calic and E. Izquierdo, “A multiresolution technique for video indexing and retrieval,” in *IEEE International Conference on Image Processing (ICIP 2002)*, 2002, vol. 1.

[16] N. O’Connor *et al.*, “Region and object segmentation algorithms in the QIMERA segmentation platform,” *Third Int. Work. Content-Based Multimed. Index.*, p. 1021+8, 2003.

[17] P. V. K. Borges, J. Mayer, and E. Izquierdo, “Efficient visual fire detection applied for video retrieval,” in *16th European Signal Processing Conference (EUSIPCO 2008)*, 2008.

[18] S. Vieweg, A. L. Hughes, K. Starbird, and L. Palen, “Microblogging During Two Natural Hazards Events: What Twitter May Contribute to Situational Awareness,” *Proc. 28th Int. Conf. Hum. factors Comput. Syst. - CHI ’10*, p. 1079, 2010.

[19] A. Schulz, P. Ristoski, and H. Paulheim, “I see a car crash: Real-time detection of small scale incidents in microblogs,” *Lect. Notes Comput. Sci. (including Subser. Lect. Notes Artif. Intell. Lect. Notes Bioinformatics)*, vol. 7955 LNCS, pp. 22–33, 2013.

[20] J. Yin, S. Karimi, A. Lampert, M. Cameron, B. Robinson, and R. Power, “Using social media to enhance emergency situation awareness,” in *Proceedings of the 24th International Conference on Artificial Intelligence*, 2015, pp. 4234–4238.

[21] T. Sakaki, M. Okazaki, and Y. Matsuo, “Earthquake Shakes Twitter Users: Real-time Event Detection by Social Sensors,” *Proc. 19th Int. Conf. World Wide Web*, pp. 851–860, 2010.

[22] J. Chae *et al.*, “Spatiotemporal Social Media Analytics for Abnormal Event

Detection and Examination using Seasonal-Trend Decomposition," *IEEE Conf. Vis. Anal. Sci. Technol. 2012, VAST 2012*, no. October 2012, pp. 143–152, 2016.

[23] V. Zavarella, H. Tanev, R. Steinberger, and E. Van Der Goot, "An Ontology-Based Approach to Social Media Mining for Crisis Management," *SMILE. Work. Soc. Media Linked Data Emerg. Response*, 2014.

[24] S. Liu, D. Shaw, and C. Brewster, "Ontologies for crisis management: A review of state of the art in ontology design and usability," *ISCRAM 2013 - 10th Int. Conf. Inf. Syst. Cris. Response Manag.*, no. May, pp. 349–359, 2013.

[25] E. Kalemi and S. Yildirim-Yayilgan, "Ontologies for Social Media Digital Evidence," *Int. J. Comput. Electr. Autom. Control Inf. Eng.*, vol. 10, no. 2, pp. 369–374, 2016.

[26] P. Thakor and S. Sasi, "Ontology-based Sentiment Analysis Process for Social Media Content," *Procedia Comput. Sci.*, vol. 53, no. 1, pp. 199–207, 2015.

[27] R. Alt and M. Wittwer, "Towards an ontology-based approach for social media analysis," in *ECIS 2014 Proceedings - 22nd European Conference on Information Systems*, 2014, pp. 1–10.

[28] X. Gao, W. Yu, Y. Rong, and S. Zhang, "Ontology-Based Social Media Analysis for Urban Planning," in *2017 IEEE 41st Annual Computer Software and Applications Conference (COMPSAC)*, 2017, no. July, pp. 888–896.

[29] E. M. Tapia, T. Choudhury, and M. Philipose, "Building Reliable Activity Models Using Hierarchical Shrinkage and Mined Ontology," *Lect. Notes Comput. Sci.*, vol. 3968, pp. 17–32, 2006.

[30] U. Akdemir, P. Turaga, and R. Chellappa, "An Ontology based Approach for Activity Recognition from Video," in *Proceedings of the 16th ACM International Conference on Multimedia*, 2008, pp. 709–712.

[31] Y. Zhang, W. Liu, N. Ding, X. Wang, and Y. Tan, "An Event Ontology Description Framework Based on SKOS," in *2015 IEEE 12th Intl Conf on Ubiquitous Intelligence and Computing and 2015 IEEE 12th Intl Conf on Autonomic and Trusted Computing and 2015 IEEE 15th Intl Conf on Scalable Computing and Communications and Its Associated Workshops (UIC-ATC-ScalCom)*, 2015, pp. 1774–1779.

[32] R. Nevatia, J. Hobbs, and B. Bolles, “An Ontology for Video Event Representation,” in *2004 Conference on Computer Vision and Pattern Recognition Workshop*, 2004, pp. 1–10.

[33] S. A. A. Mappe and P. Wongthongtham, “Knowledge Representation for Potential Field of Study Recognition,” in *2013 IEEE International Conference on Teaching, Assessment and Learning for Engineering (TALE)*, 2013, no. August, pp. 160–163.

[34] C. Guérin, C. Rigaud, K. Bertet, and A. Revel, “An ontology-based framework for the automated analysis and interpretation of comic books’ images,” *Inf. Sci. (Ny).*, vol. 378, pp. 109–130, 2017.

[35] J. A. Bullinaria, “IAI : Knowledge Representation What is Knowledge ?,” 2005, pp. 1–20.

[36] F. van Harmelen, V. Lifschitz, and B. Porter, *Handbook of Knowledge Representation*. 2008.

[37] P. Bernus, J. Blazewicz, S. Gunter, and M. Shaw, *Handbook on Ontologies*. 2009.

[38] M. Bertini, A. Del Bimbo, and G. Serra, “Learning ontology rules for semantic video annotation,” in *Proc of ACM International Conference on Multimedia Many Faces of Multimedia Semantics MS*, 2008, pp. 1–8.

[39] A. Abraham, *Rule-based Expert Systems*. 2005.

[40] F. Baader, D. Calvanese, D. L. McGuinness, D. Nardi, and P. F. Patel-Schneider, “The Description Logic Handbook: Theory, Implementation and Applications,” *Kybernetes*, vol. 32, no. 9/10, p. 624, 2010.

[41] F. Baader, I. Horrocks, and U. Sattler, “Description Logics,” in *Handbook of Knowledge Representation*, 2008, pp. 135–180.

[42] D. Calvanese, G. De Giacomo, M. Lenzerini, and D. Nardi, “Reasoning in Expressive Description Logics,” *Handb. Autom. Reason.*, pp. 1581–1634, 2001.

[43] W. N. Borst, “Construction of Engineering Ontologies for Knowledge Sharing and Reuse,” 1997.

[44] F. Bobillo and U. Straccia, “Fuzzy ontology representation using OWL 2,” *Int. J. Approx. Reason.*, vol. 52, no. 7, pp. 1073–1094, Oct. 2011.

[45] D. Jones, T. Bench-Capon, and P. Visser, “Methodologies For Ontology Development,” in *In Proc. IT&KNOWS Conference of the 15th IFIP World*

Computer Congress, 1998.

[46] H. S. Pinto and J. P. Martins, “Ontologies: How can They be Built?,” *Knowl. Inf. Syst.*, vol. 6, pp. 441–464, 2004.

[47] I. A. Al-Baltah, A. A. A. Ghani, W. N. W. A. Rahman, and R. Atan, “A comparative study on ontology development methodologies towards building semantic conflicts detection ontology for heterogeneous web services,” *Res. J. Appl. Sci. Eng. Technol.*, vol. 7, no. 13, pp. 2674–2679, 2014.

[48] M. Uschold and M. King, “Towards a Methodology for Building Ontologies,” *Work. Basic Ontol. Issues Knowl. Shar.*, vol. 80, no. July, pp. 275–280, 1995.

[49] F. M. López, “Overview Of Methodologies For Building Ontologies,” in *Proceedings of the IJCAI99 Workshop on Ontologies and ProblemSolving Methods Lessons Learned and Future Trends CEUR Publications*, 1999, no. 2, pp. 1–13.

[50] M. Gruninger and M. S. Fox, “Methodology for the Design and Evaluation of Ontologies,” in *Workshop on Basic Ontological Issues in Knowledge Sharing, IJCAI-95*, 1995.

[51] A. De Nicola, M. Missikoff, and R. Navigli, “A software engineering approach to ontology building,” *Inf. Syst.*, vol. 34, no. 2, pp. 258–275, 2009.

[52] M. Fernández, A. Gómez-Pérez, and N. Juristo, “METHONTOLOGY: From Ontological Art Towards Ontological Engineering,” *AAAI-97 Spring Symp. Ser.*, vol. SS-97-06, pp. 33–40, 1997.

[53] G. Brusa, M. L. Caliusco, and O. Chiotti, “Towards Ontological Engineering: A Process for Building a Domain Ontology from Scratch in Public Administration,” *Expert Syst.*, vol. 25, no. 5, pp. 484–503, 2008.

[54] M. Fernández-López, A. Gómez-Pérez, and M. D. Rojas, “Ontology’s crossed life cycles,” in *International Conference on Knowledge Engineering and Knowledge Management*, 2000, pp. 155–179.

[55] A. Saad and S. Shaharin, “The Methodology for Ontology Development in Lesson Plan Domain,” *Int. J. Adv. Comput. Sci. Appl.*, vol. 7, no. 4, pp. 557–562, 2016.

[56] M. C. Suárez-figueroa and A. Gómez-pérez, “How to Write and Use the Ontology Requirements Specification Document,” no. November 2009,

2019.

[57] G. Guizzardi and G. Wagner, “A Unified Foundational Ontology and some Applications of it in Business Modeling,” in *CAiSE Workshops*, 2004, pp. 129–143.

[58] C. M. Keet, “The Use of Foundational Ontologies in Ontology Development: An Empirical Assessment,” in *The Semantic Web: Research and Applications. ESWC 2011. Lecture Notes in Computer Science*, 2011, vol. 6643 LNCS, no. PART 1, pp. 321–335.

[59] V. Mascardi, V. Cordì, and P. Rosso, “A Comparison of Upper Ontologies,” in *WOA*, 2007, no. May, pp. 55–64.

[60] C. Masolo, S. Borgo, A. Gangemi, N. Guarino, A. Oltramari, and I. Horrocks, “WonderWeb Deliverable D18 Ontology Library (final),” 2003.

[61] C. Roussey, F. Pinet, M. A. Kang, and O. Corcho, “An Introduction to Ontologies and Ontology Engineering,” pp. 9–38, 2011.

[62] S. Shantaiya, K. Verma, and K. Mehta, “A Survey on Approaches of Object Detection,” *Int. J. Comput. Appl.*, vol. 65, no. 18, pp. 14–20, 2013.

[63] P. Viola and M. Jones, “Rapid Object Detection using a Boosted Cascade of Simple Features,” in *Proceedings of the 2001 IEEE Computer Society Conference on Computer Vision and Pattern Recognition, CVPR*, 2001, vol. 1, pp. 511–518.

[64] N. Dalal and B. Triggs, “Histograms of Oriented Gradients for Human Detection,” *IEEE Comput. Soc. Conf. Comput. Vis. Pattern Recognition, (CVPR 2005)*, vol. 1, pp. 886–893, 2005.

[65] P. Dollár, Z. Tu, P. Perona, and S. Belongie, “Integral Channel Features,” *Proc. Br. Mach. Vis. Conf. 2009*, pp. 1–11, 2009.

[66] P. Viola, O. M. Way, M. J. Jones, and D. Snow, “Detecting Pedestrians Using Patterns of Motion and Appearance,” *Int. J. Comput. Vis.*, vol. 63, no. 2, pp. 153–161, 2005.

[67] P. Viola and M. J. Jones, “Robust Real-Time Face Detection,” *Int. J. Comput. Vis.*, vol. 57, no. 2, pp. 137–154, 2004.

[68] “Guide to Authoring Media Ground Truth with ViPER-GT,” 2003. [Online]. Available: http://viper-toolkit.sourceforge.net/docs/gt/4.0/tutorial/. [Accessed: 01-Apr-2018].

[69] D. Mihalcik and D. Doermann, *The Design and Implementation of ViPER*.

2003.

[70] H. Cunningham *et al.*, *Developing language processing components with GATE Version 8*, vol. 8. 2017.

[71] S. Decker *et al.*, “The Semantic Web: The Roles of XML and RDF,” *IEEE Internet Comput.*, vol. 4, no. October, pp. 63–74, 2000.

[72] N. Bikakis, C. Tsinaraki, N. Gioldasis, I. Stavrakantonakis, and S. Christodoulakis, “The XML and Semantic Web Worlds: Technologies, Interoperability and Integration. A survey of the State of the Art,” in *Semantic Hyper/Multimedia Adaptation: Schemes and Applications*, vol. 418, 2013, pp. 3–14.

[73] S. Bechhofer *et al.*, “OWL Web Ontology Language Reference,” 2004.

[74] G. Madhu and A. G. T. V Rajinikanth, “Intelligent Semantic Web Search Engines : A Brief Survey,” *Int. J. Web Semant. Technol.*, vol. 2, no. 1, pp. 34–42, 2011.

[75] L. Quin, “Extensible Markup Language (XML),” *W3C*, 2016. [Online]. Available: https://www.w3.org/XML/. [Accessed: 14-May-2018].

[76] C. D. Manning, P. Raghavan, and H. Schutze, *Introduction to Information Retrieval*. Cambridge University Press Cambridge, UK, 2008.

[77] M. Hacherouf, S. N. Bahloul, and C. Cruz, “Transforming XML documents to OWL ontologies: A survey,” *J. Inf. Sci.*, vol. 41, no. 2, pp. 242–259, 2015.

[78] M. Kay, “XSL Transformations (XSLT) Version 3.0,” 2017. [Online]. Available: https://www.w3.org/TR/2017/REC-xslt-30-20170608/. [Accessed: 22-Feb-2018].

[79] “Resource Description Framework (RDF),” *W3C*, 2014. [Online]. Available: https://www.w3.org/RDF/. [Accessed: 29-Mar-2018].

[80] “Web Ontology Language (OWL),” *W3C*, 2012. [Online]. Available: https://www.w3.org/OWL/.

[81] H. Zhuge, Y. Xing, and P. Shi, “Resource space model, OWL and database,” *ACM Trans. Internet Technol.*, vol. 8, no. 4, pp. 1–31, 2008.

[82] P. Hitzler, M. Krötzsch, B. Parsia, P. F. Patel-Schneider, and S. Rudolph, “OWL 2 Web Ontology Language Primer,” 2009.

[83] M. Ferdinand, C. Zirpins, and D. Trastour, “Lifting XML Schema to OWL,” *Web Eng. ICWE 2004. Lect. Notes Comput. Sci.*, no. 3140, pp. 354–358,

2004.

[84] H. Bohring and S. Auer, “Mapping XML to OWL Ontologies,” *Leipziger Inform.*, vol. 72, pp. 147–156, 2005.

[85] C. Tsinaraki and S. Christodoulakis, “Interoperability of XML Schema Applications with OWL Domain Knowledge and Semantic Web Tools,” *Lect. Notes Comput. Sci.*, vol. 4803, pp. 850–869, 2007.

[86] C. Cruz and C. Nicolle, “Ontology Enrichment and Automatic Population From XML Data,” *OBDIS*, pp. 17–20, 2008.

[87] R. Ghawi and N. Cullot, “Building Ontologies from XML Data Sources,” *Proc. - Int. Work. Database Expert Syst. Appl. DEXA*, pp. 480–484, 2009.

[88] I. Bedini, C. Matheus, P. F. Patel-Schneider, A. Boran, and B. Nguyen, “Transforming XML schema to OWL using patterns Transforming XML Schema to OWL Using Patterns,” *IEEE Fifth Int. Conf. Semant. Comput.*, pp. 102–109, 2011.

[89] D. Lacoste, K. P. Sawant, and S. Roy, “An efficient XML to OWL converter,” *Proc. 4th India Softw. Eng. Conf. 2011, ISEC’11*, pp. 145–154, 2011.

[90] N. Yahia, S. A. Mokhtar, and A. Ahmed, “Automatic Generation of OWL Ontology from XML Data Source,” *Int. J. Comput. Sci. Issues*, vol. 9, no. 2, pp. 77–83, 2012.

[91] C. Ramirez and B. Valdes, “A General Knowledge Representation Model of Concepts,” in *Advances in Knowledge Representation*, no. May, 2012, p. 272.

[92] M. Mrak, N. Sprljan, T. Zgaljic, N. Ramzan, S. Wan, and E. Izquierdo, “Performance evidence of software proposal for Wavelet Video Coding Exploration Group,” 76th MPEG Meeting, Montreux, Switzerland, 2006.

[93] M. Bais *et al.*, “Customized television: Standards compliant advanced digital television,” *IEEE Trans. Broadcast.*, vol. 48, no. 2, pp. 151–158, 2002.

[94] N. Sprljan, M. Mrak, G. C. K. Abhayaratne, and E. Izquierdo, “A Scalable Coding Framework for Efficient Video Adaptation,” *6th Int. Work. Image Anal. Multimed. Interact. Serv. (WIAMIS 2005)*, pp. 1–4, 2005.

[95] J. Calic and E. Izquierdo, “Temporal Segmentation of MPEG Video Streams,” *EURASIP J. Adv. Signal Process.*, vol. 2002, no. 6, pp. 561–

565, 2002.

[96] K. Chandramouli, T. Kliegr, J. Nemrava, V. Svatek, and E. Izquierdo, “Query refinement and user relevance feedback for contextualized image retrieval,” in *5th International Conference on Visual Information Engineering (VIE 2008)*, 2008, no. 543 CP, pp. 453–458.

[97] Q. Zhang and E. Izquierdo, “Combining Low-Level Features for Semantic Extraction in Image Retrieval,” *EURASIP J. Adv. Signal Process.*, vol. 2007, no. 1, p. 12, 2007.

[98] Y. K. Tani, A. Lablack, A. Ghomari, and I. M. Bilasco, “Events detection using a video-surveillance ontology and a rule-based approach,” *Lect. Notes Comput. Sci.*, vol. 8926, pp. 299–308, 2015.

[99] K. Dilek Onal and P. Karagoz, “Named Entity Recognition from Scratch on Social Media,” in *Proceedings of the 6th International Conference on Mining Ubiquitous and Social Environments (MUSE’15)*, 2015, vol. 1521, pp. 2–17.

[100] A. Bhoi and R. C. Balabantaray, “Named Entity Recognition from Social Media Text : A Comparative Study,” *Int. J. Control Theory Appl.*, vol. 10, no. 19, pp. 9–15, 2017.

[101] A. P. Kumar, K. Abhishek, and V. K. N, “Architecting and Designing of Semantic Web Based Application using the JENA and PROTÉGÉ – A Comprehensive Study,” *Histograms Oriented Gradients Hum. Detect.*, vol. 2, no. 3, pp. 1279–1282, 2011.

[102] H. Andreas, M. Janik, and S. Staab, *Semantic Web Architecture*. 2011.

[103] F. Breitling, “A standard transformation from XML to RDF via XSLT,” *Astron. Nachrichten*, vol. 330, no. 7, pp. 755–760, 2009.

[104] K. Dentler, R. Cornet, A. Ten Teije, and N. De Keizer, “Comparison of reasoners for large ontologies in the OWL 2 EL profile,” *Semant. Web*, vol. 2, no. 2, pp. 71–87, 2011.

[105] S. Dosis, I. Homem, and O. Popov, “Semantic representation and integration of digital evidence,” *Procedia Comput. Sci.*, vol. 22, pp. 1266–1275, 2013.

[106] V. Fortineau, T. Paviot, L. Louis-Sidney, and S. Lamouri, “SWRL as a rule language for ontology-based models in power plant design,” *IFIP Adv. Inf. Commun. Technol.*, vol. 388 AICT, pp. 588–597, 2012.

[107] A. Ayadi, C. Zanni-Merk, F. D. B. De Beuvron, and S. Krichen, “Ontological reasoning for understanding the behaviour of complex biomolecular networks,” in *Proceedings of IEEE/ACS International Conference on Computer Systems and Applications, AICCSA*, 2017, vol. 2017–Octob, pp. 1486–1493.

[108] S. Han, A. Hutter, and W. Stechele, “Toward Contextual Forensic Retrieval for Visual Surveillance : Challenges and an Architectural Approach,” in *2009 10th Workshop on Image Analysis for Multimedia Interactive Services*, 2009, pp. 201–204.

[109] E. Sirin, B. Parsia, B. C. Grau, A. Kalyanpur, and Y. Katz, “Pellet: A practical OWL-DL reasoner,” *Web Semant. Sci. Serv. Agents World Wide Web*, vol. 5, no. 2, pp. 51–53, 2007.

[110] I. Horrocks, U. Sattler, and S. Tobies, “Practical Reasoning for Expressive Description Logics,” in *Logic for Programming and Automated Reasoning*, 1999, pp. 161–180.

[111] D. Vrande *et al.*, “Ontology Evaluation,” *Handb. Ontol.*, vol. 16, no. June, pp. 251–274, 2010.

[112] F. Sobhani, N. F. Kahar, and Q. Zhang, “An ontology framework for automated visual surveillance system,” in *Content-Based Multimedia Indexing (CBMI), 2015 13th International Workshop on*, 2015, pp. 1–7.

[113] J. Gómez Romero, “A Practical Approach to the Development of Ontology-Based Information Fusion Systems,” pp. 176–183, 2013.

[114] L. Ballan, M. Bertini, A. Del Bimbo, L. Seidenari, and G. Serra, “Event detection and recognition for semantic annotation of video,” *Multimed. Tools Appl.*, vol. 51, no. 1, pp. 279–302, 2010.

[115] U. Westermann and R. Jain, “Toward a common event model for multimedia applications,” *IEEE Multimed.*, vol. 14, no. 1, pp. 19–29, 2007.

[116] A. Quinton, “Objects and events,” *Mind*, vol. 88, no. 1, pp. 197–214, 1979.

[117] E. Itkonen, “Causality in Linguistic Theory. A Critical Investigation into the Philosophical and Methodological Foundations of Non-Autonomous Linguistics,” *Int. J. Spons. by Found. “Foundations Lang.*, vol. 10, no. 1, pp. 208–213, 1986.

[118] Y. Sun and Z. Li, “Ontology-based domain knowledge representation,” in *4th International Conference on Computer Science & Education*, 2009, pp. 174–177.

[119] A. Scherp, T. Franz, C. Saathoff, and S. Staab, “A core ontology on events for representing occurrences in the real world,” *Multimed. Tools Appl.*, vol. 58, no. 2, pp. 293–331, 2012.

[120] A. Gangemi, N. Guarino, C. Masolo, A. Oltramari, and L. Schneider, “Sweetening Ontologies with DOLCE,” *Knowl. Eng. Knowl. Manag. Ontol. Semant. Web. EKAW 2002*, vol. 2473, 2002.

[121] A. Gangemi, N. Guarino, C. Masolo, and A. Oltramari, “Sweetening WORDNET with DOLCE,” *AI Mag.*, vol. 24, no. 3, pp. 13–24, 2003.

[122] J. Hartmann, P. Spyns, A. Giboin, and D. Maynard, *Methods for Ontology Evaluation*. 2005.

[123] H. Y. Wang, “London Crime Data - EDA,” 2017. [Online]. Available: https://www.kaggle.com/sw52099/london-crime-data-eda/data.

[124] “Structural Specification and Functional-Style Syntax (Second Edition),” *W3C*, 2009. [Online]. Available: https://www.w3.org/2007/OWL/wiki/Syntax. [Accessed: 21-Mar-2019].

[125] M. M. Kokar, C. J. Matheus, and K. Baclawski, “Ontology-based situation awareness,” *Inf. fusion*, vol. 10, no. 1, pp. 83–98, 2009.

[126] S. Harris and A. Seaborne, “SPARQL 1.1 Query Language,” *W3C*, 2013. [Online]. Available: https://www.w3.org/TR/sparql11-query/.

[127] B. Parr, “London Riots: Twitter Traffic Surges in the UK,” 2011. [Online]. Available: https://mashable.com/2011/08/09/london-riots-twitter/#KnRttvORz5qp. [Accessed: 10-Mar-2018].

[128] V. S. Silva, A. Freitas, and S. Handschuh, “Word tagging with foundational ontology classes: Extending the WordNet-DOLCE mapping to verbs,” *Lect. Notes Comput. Sci. (including Subser. Lect. Notes Artif. Intell. Lect. Notes Bioinformatics)*, vol. 10024 LNAI, pp. 593–605, 2016.

[129] J. Brownlee, “Welcome to Deep Learning,” 2016.

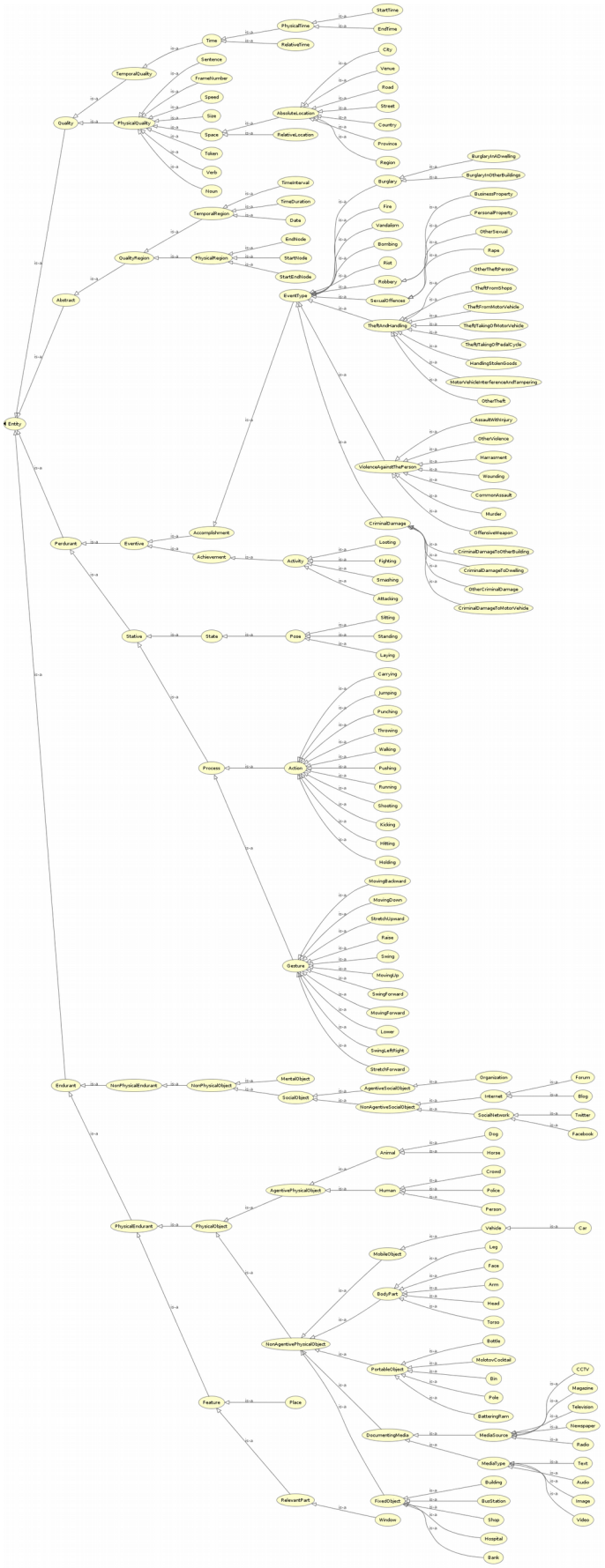
[130] “What is a Graph Database?,” *neo4j*. [Online]. Available: https://neo4j.com/developer/graph-database/#property-graph. [Accessed: 31-Mar-2019].

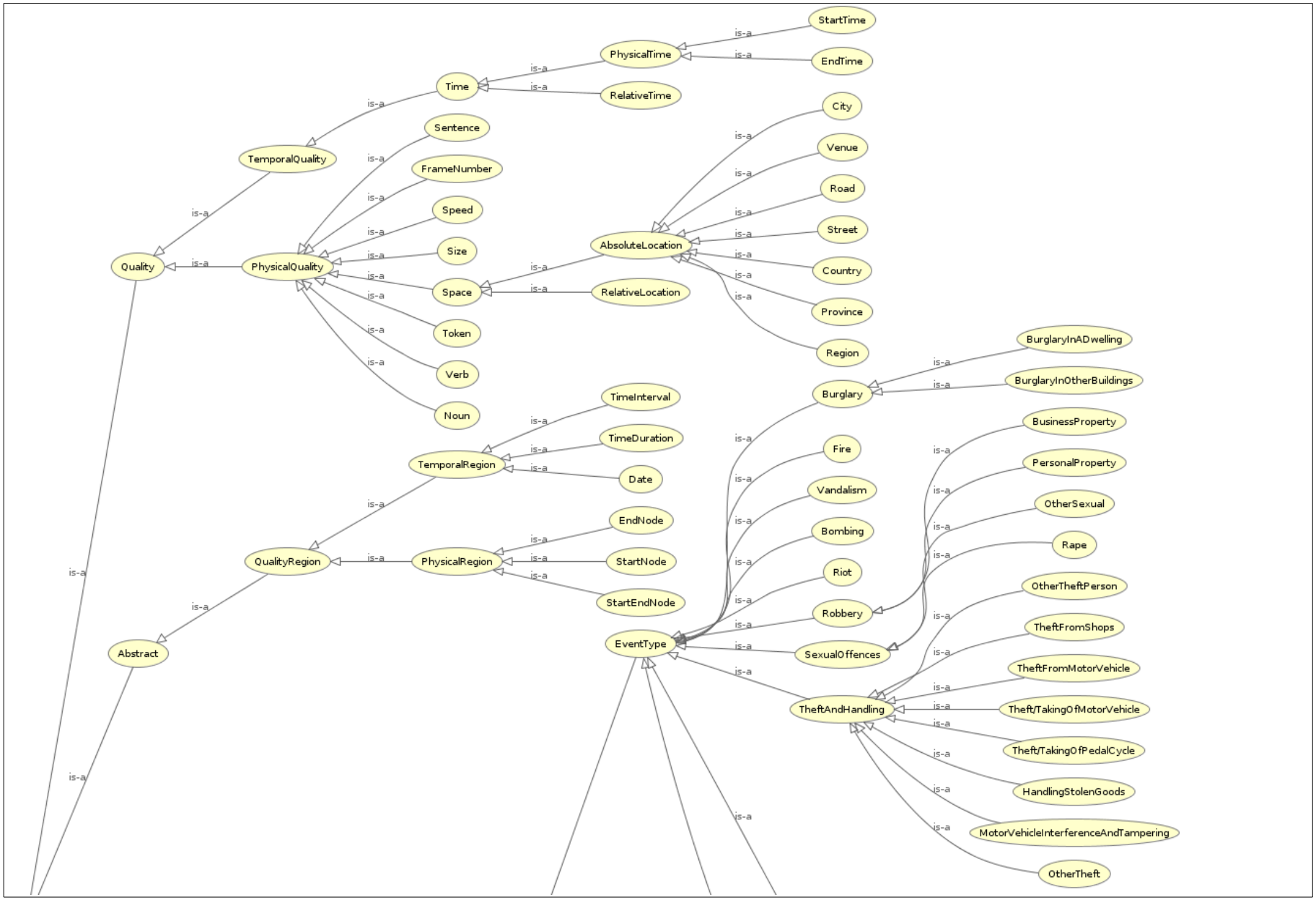
[131] M. Conway, J. Dowling, and W. Chapman, “Developing a Biosurveillance Application Ontology for,” no. August, pp. 58–66, 2010.

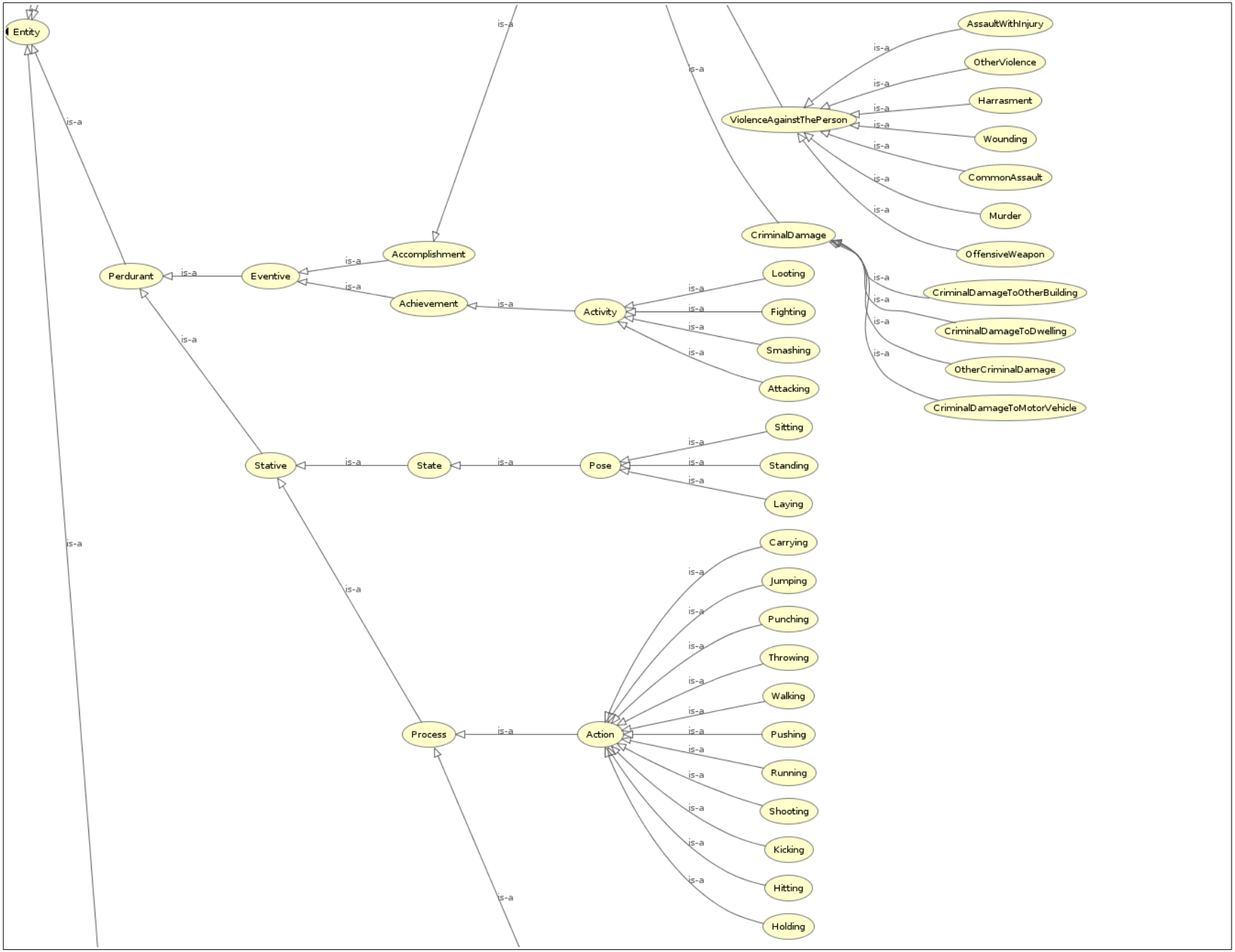
APPENDIX

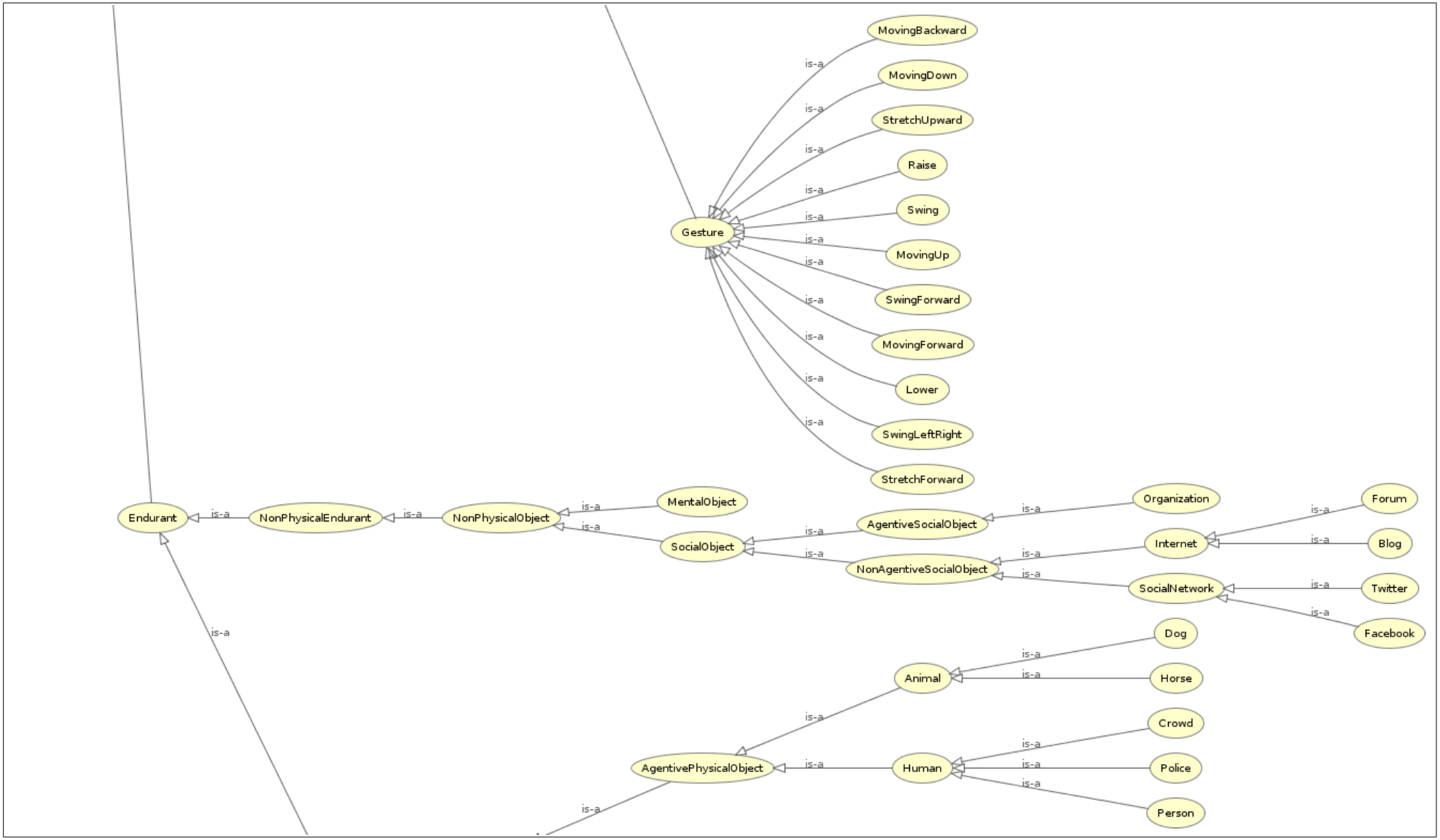
SECURITY DOMAIN ONTOLOGY

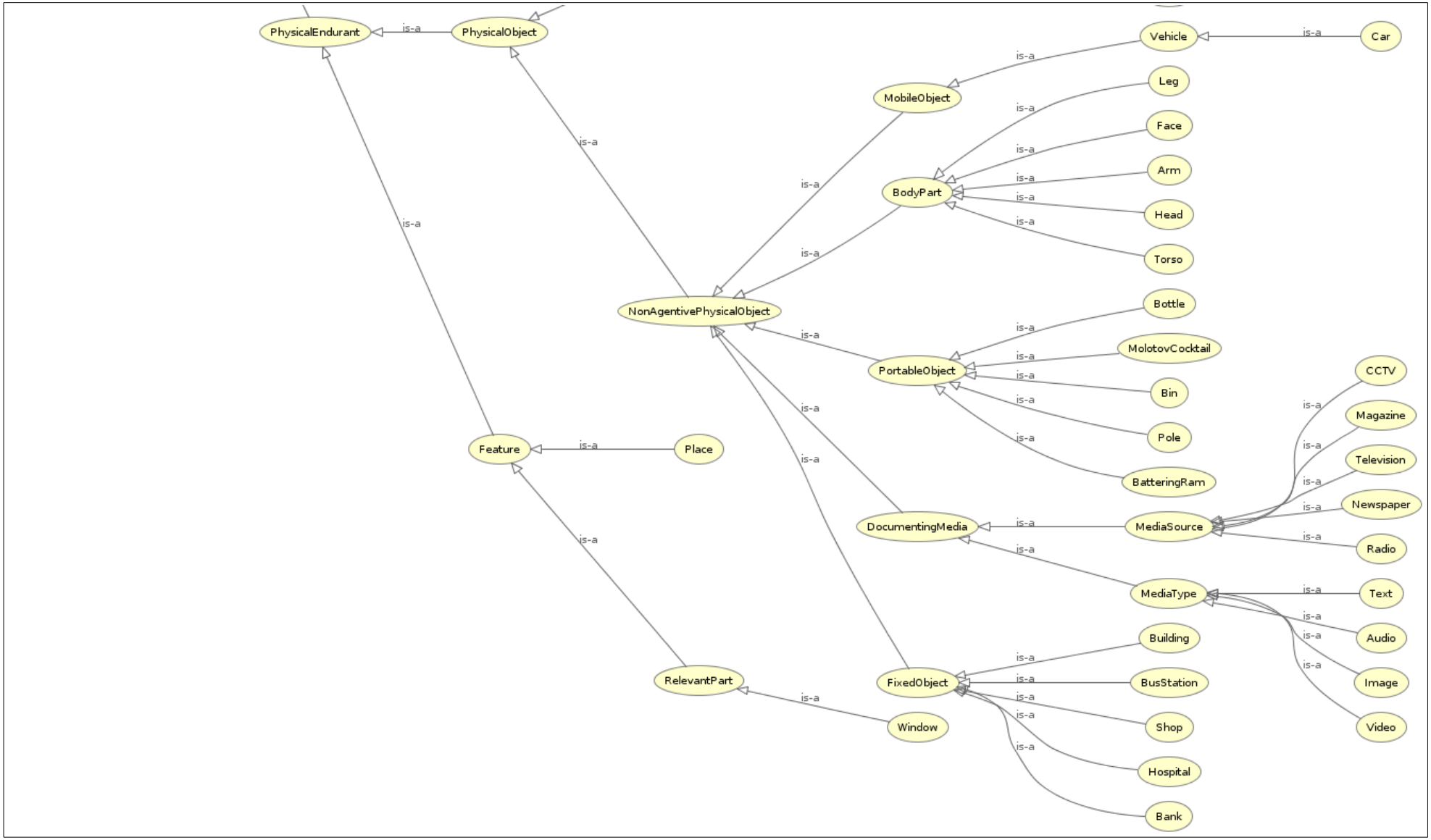
A complete illustration of the security domain ontology that has been elaborated in Chapter 4 is presented here. The security domain ontology concept hierarchy consists of 174 classes, 23 object properties, 38 data properties and 490 axioms. These includes DOLCE's core categories of particulars enduring, perduring, abstract and quality.


Entity
Abstract
Quality
Time
PhysicalTime
StartTime
EndTime
RelativeTime
Sentence
TemporalQuality
FrameNumber
PhysicalQuality
Speed
Size
Space
Token
Verb
Noun
AbsoluteLocation
City
Venue
Road
Street
Country
Province
Region
RelativeLocation
QualityRegion
TemporalRegion
TimeInterval
TimeDuration
Date
PhysicalRegion
EndNode
StartNode
StartEndNode
EventType
Fire
Vandalism
Bombing
Riot
Robbary
SexualOffences
Burglary
BurglaryInADwelling
BurglaryInOtherBuildings
BusinessProperty
PersonalProperty
OtherSexual
Rape
OtherTheftPerson
TheftFromShops
TheftFromMotorVehicle
TheftHandling
TheftTakingOfMotorVehicle
TheftTakingOfPedalCycle
HandlingStolenGoods
MotorVehicleInterferenceAndTampering
OtherTheft
ViolenceAgainstThePerson
AssaultWithInjury
OtherViolence
Harrassment
Wounding
CommonAssault
Murder
OffensiveWeapon
CriminalDamage
CriminalDamageToOtherBuilding
CriminalDamageToDwelling
OtherCriminalDamage
CrimnalDamageToMotorVehicle
Perdurant
Eventive
Accomplishment
Achievement
Activity
Looting
Fighting
Smashing
Attacking
Stative
State
Pose
Sitting
Standing
Laying
Process
Action
Carrying
Jumping
Punching
Throwing
Walking
Pushing
Running
Shooting
Kicking
Hitting
Holding
Gesture
MovingBackward
MovingDown
StretchUpward
Raise
Swing
MovingUp
SwingForward
MovingForward
Lower
SwingLeftRight
StretchForward
Endurant
NonPhysicalEndurant
NonPhysicalObject
MentalObject
SocialObject
AgentiveSocialObject
Organization
NonAgentiveSocialObject
Internet
Forum
Blog
SocialNetwork
Twitter
Facebook
PhysicalEndurant
PhysicalObject
AgentivePhysicalObject
Animal
Dog
Horse
Human
Crowd
Police
Person
NonAgentivePhysicalObject
MobileObject
Vehicle
Car
BodyPart
Leg
Face
Arm
Head
Torso
PortableObject
Bottle
MolotovCocktail
Bin
Pole
BatteringRam
Feature
Place
RelevantPart
DocumentingMedia
MediaSource
CCTV
Magazine
Television
Newspaper
Radio
MediaType
Text
Audio
Image
Video
FixedObject
Building
BusStation
Shop
Hospital
Bank
Window


StartTime
PhysicalTime
EndTime
Time
RelativeTime
TemporalQuality
Sentence
FrameNumber
Speed
Size
Quality
PhysicalQuality
Space
Token
Verb
Noun
City
Venue
Road
Street
AbsoluteLocation
Country
Province
Region
RelativeLocation
BurglaryInADwelling
BurglaryInOtherBuildings
Burglary
TimeInterval
TimeDuration
TemporalRegion
Date
Fire
Vandalism
Bombing
Riot
Robbery
SexualOffences
EventType
BusinessProperty
PersonalProperty
OtherSexual
Rape
OtherTheftPerson
TheftFromShops
TheftFromMotorVehicle
Theft/TakingOfMotorVehicle
Theft/TakingOfPedalCycle
HandlingStolenGoods
MotorVehicleInterferenceAndTampering
OtherTheft
TheftAndHandling
EndNode
StartNode
StartEndNode
QualityRegion
PhysicalRegion
Abstract
is-a


Entity
is-a
Perdurant
Eventive
Accomplishment
Achievement
Activity
Looting
Fighting
Smashing
Attacking
Stative
State
Pose
Sitting
Standing
Laying
Process
Action
Carrying
Jumping
Punching
Throwing
Walking
Pushing
Running
Shooting
Kicking
Hitting
Holding
ViolenceAgainstThePerson
AssaultWithInjury
OtherViolence
Harrasment
Wounding
CommonAssault
Murder
OffensiveWeapon
CriminalDamage
CriminalDamageToOtherBuilding
CriminalDamageToDwelling
OtherCriminalDamage
CriminalDamageToMotorVehicle


MovingBackward
MovingDown
StretchUpward
Raise
Swing
MovingUp
SwingForward
MovingForward
Lower
SwingLeftRight
StretchForward
Gesture
is-a
Endurant
NonPhysicalEndurant
NonPhysicalObject
MentalObject
SocialObject
AgentiveSocialObject
NonAgentiveSocialObject
Organization
Internet
SocialNetwork
Forum
Blog
Twitter
Facebook
Dog
Horse
Animal
Crowd
Police
Person
Human
AgentivePhysicalObject


PhysicalEndurant
is-a
PhysicalObject
Vehicle
is-a
Car
MobileObject
Leg
Face
Arm
Head
Torso
BodyPart
NonAgentivePhysicalObject
Bottle
MolotovCocktail
Bin
Pole
BatteringRam
PortableObject
CCTV
Magazine
Television
Newspaper
Radio
MediaSource
DocumentingMedia
MediaType
Text
Audio
Image
Video
Feature
Place
RelevantPart
Window
Building
BusStation
Shop
Hospital
Bank
FixedObject