Detecting Riots with Uncertain Information on the Semantic Web

Cesar Pantoja

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

First Supervisor: Prof. Ebroul Izquierdo
Second Supervisor: Dr. Qianni Zhang

PhD Thesis

School of of Electronic Engineering and Computer Science
Queen Mary University of London
United Kingdom

September 2016

Statement of Originality

I, Cesar Hernando Pantoja Sanchez, 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: 29th of October, 2016

Details of collaboration and publications:

All papers published while working on this thesis are listed at the end of the thesis. Any publication produced in collaboration with others is clearly mentioned.

Abstract

The ubiquitous nature of CCTV Surveillance cameras means substantial amounts of data being generated. In case of an investigation, this data must be manually browsed and analysed in search of relevant information for the case. As an example, it took more than 450 detectives to examine the hundreds of thousands of hours of videos in the investigation of the 2011 London Riots: one of the largest the London's MET police has ever seen. Anything that can help the security forces save resources in investigations such as this, is valuable. Consequently, automatic analysis of surveillance scenes is a growing research area.

One of the research fronts tackling this issue, is the semantic understanding of the scene. In this, the output of computer vision algorithms is fed into Semantic Frameworks, which combine all the information from different sources and try to reach a better knowledge of the scene. However, representing and reasoning with imprecise and uncertain information remains an outstanding issue in current implementations.

The Demspter-Shaffer (DS) Theory of Evidence has been proposed as a way to deal with imprecise and uncertain information. In this thesis we use it for the main contributions.

In our first contribution, we propose the use of the DS theory and its Transferable Belief Model (TBM) realisation as a way to combine Bayesian *priors*, using the subjectivist view of the Bayes' Theorem, where the probabilities are beliefs. We first compute the *a priori* probabilities of all the pair of events in the model. Then a global potential is created for each event using the TBM. This global potential will encode all the *prior* knowledge for that particular concept. This has the benefit that when this potential is included in a knowledge base because it has been learned, all the knowledge it entails comes with it.

We also propose a semantic web reasoner based on the TBM. This reasoner consists of an ontology to model any domain knowledge using the TBM constructs of Potentials, Focal Elements, and Configurations. The reasoner also consists of the implementations of the TBM operations in a semantic web framework. The goal is that after the model has been created, the TBM operations can be applied and the knowledge combined and queried. These operations are computationally complex, so we also propose parallel heuristics to the TBM operations. This allows us to apply this paradigm on problems of thousands of records.

The final contribution, is the use of the TBM semantic framework with the method to combine the *prior* knowledge to detect riots on CCTV footage from the 2011 London riots. We use around a million and a half manually annotated frames with 6 different concepts related to the riot detection task, train the system, and infer the presence of riots in the test dataset. Tests show that the system yields a high recall, but a low precision, meaning that there are a lot of false positives. We also show that the framework scales well as more compute power becomes available.

Acknowledgments

> "If you are doing what no one has done before, stuff goes wrong. And in fact, if nothing ever goes wrong [...] then you are not on the frontier. Simple."
>
> *Neil deGrasse Tyson*

The work presented here could not have been done without the guidance from my supervisor, Ebroul Izquierdo. I am grateful for the opportunity he gave me, and for his support and patience during difficult times. It has been an honour to be your student. I would also like to thank my second supervisor, Qianni Zhang, who also advised me during crucial work of this PhD.

I am also forever thankful of María back in Cali, who helped me secure the PhD position in the first place, and who also gave me her support and words of encouragement during my whole time here.

This is also an achievement for Krishna, Tomas, Fiona, and Virginia at MMV, who were also always there in times of need, and were always willing to lend me a hand, an ear, and much more. Of course I would like to thank all my friends at MMV, at QMUL, and in London and everywhere in the world, who also made my experience an unforgettable one. You are too many to name, but you know who you are and will always be in my heart.

My family also played an important role in this achievement. I would like to thank My Uncle Enrique, who also supported me consistently and without question during my time here. I will be forever thankful. My sister Diana, and Jeroen, who helped me in so many

ways and will forever be in debt with them. My mum was also crucial, and I would like to thank her for making me the person I am now. Diana, the mother of my daughters, who has been very patient during this time and for all the love and good values she gives to our babies.

And finally, my baby girls, Sara and Sofía, who have been since the day their were born, a source of inspiration and strength. They have given me the discipline to achieve my dreams, they have been my friends, my company, my everything. I want you to know that I miss you a lot, and can not wait until we are together again.

Table of Contents

List of Figures

Chapter 1

Introduction

High crime rates lead to large volumes of police cases, each with its own set of evidence and facts. Currently, this evidence must be browsed manually by agents or security forces to find even minor connections that lead to possible solutions to the cases. This leads to a potential risk of information overload on the operators, because of the large amount of information available and because existing systems focus on presenting the available information on graphical user interfaces (GUIs), but not actually helping the user discover knowledge within the existing information. Smart Surveillance Systems are a real alternative to assist the jobs of security forces. Systems that not only present existing information but actually help the operator discover knowledge.

As an example, on August of 2011, England was swept by a wave of riots for four days in some of its cities (including a few locations in London) causing an estimated £200 million in damages [4]. In London alone, where at the time there was an estimated 8,000 CCTV cameras in and around the city [5], it took more than 450 detectives to examine the evidence, including hundreds of thousands of hours of footage [6] [7] in the resulting investigation. This concluded in about 5,000 perpetrators arrested, some 4,000 of those thanks to CCTV footage [8].

One of the main areas of research in the surveillance domain is the semantic under-

standing of CCTV footage. Widespread use of multimedia capture devices such as smart-phones also poses the challenge of indexing and understanding the content generated by these devices for potential use in police investigations. Recently, research on standard-based semantically enriched knowledge models for surveillance and forensic use is intensifying. They are seen as an alternative to ad-hoc systems that offer little interoperability and do not adhere to any standard.

The Semantic Web, as defined by the W3C, "provides a common framework that allows data to be shared and reused across application, enterprise, and community boundaries" [9]. Using semantic web technologies for Smart Surveillance Systems offers several advantages:

- The World Wide Web is large. It is comprised of millions upon millions of documents, hosted in an equally large network of computers. This means that any development towards the semantic web must ensure operability in such a large scale.

- Semantic Web still has an active research community tackling different issues, including modelling, indexing, querying, and reasoning; key aspects for the surveillance task.

- Standard compliance allows the easy exchange of information between different data sources, expanding the available knowledge for the reasoning and knowledge discovery task.

When dealing with information from the real world, one has to take into account the 'ignorance' inherent in the data-sources. But ignorance can take many forms. Already in the literature there have been efforts to categorise all forms of ignorance [10]. For this work we are going to use the taxonomy proposed by [11]. In summary, they define three types of ignorance:

- **Incompleteness:** In this type of ignorance, not all the information is known, but

you are certain that the information you possess is correct. The semantic web is particularly strong in modelling this kind of ignorance. For example we might know that person A has a parent B, and B has a brother C. Through rules and other reasoning methods we can infer that A has an uncle C, even though that specific piece of knowledge was not encoded in the knowledge base. This particular form of ignorance can also be handled with non-monotonic logic (like default reasoning [12], abductive reasoning [13] and backtracking).

- **Imprecision:** It is when data is available with an imprecise measurement, but you know such imprecise value to be right. For example you might know that person A likes rock music, but you are not certain of how much that person likes it (it could be a casual interest or they could be fanatics). This is particularly interesting in the field of Artificial Intelligence as classifiers and other types of algorithms return a degree of confidence when applied to real world data. It is dealt with by using *fuzzy sets*.

- **Uncertainty:** There are cases where the data might be wrong. Such is the case of for example faulty sensors. A more relevant example for the forensic domain is when a witness gives a bad testimony. *Probability theory* (see Figure 1.1) allows you to represent this in the form of the probability that the information you have is wrong.

Different forms of ignorance might manifest themselves in different domains. For example, an unreliable witness might tell you that the perpetrator of a crime was a young person. *Evidence Theory* and *Possibility Theory* have been proposed to reason with imprecision and uncertainty [1].

As stated previously, the Semantic Web is well suited to handle incomplete data. But the technical challenges of representing and reasoning with imprecise and uncertain information still remain an outstanding issue.

Bayesian Theory is a common method for dealing with probability problems, but

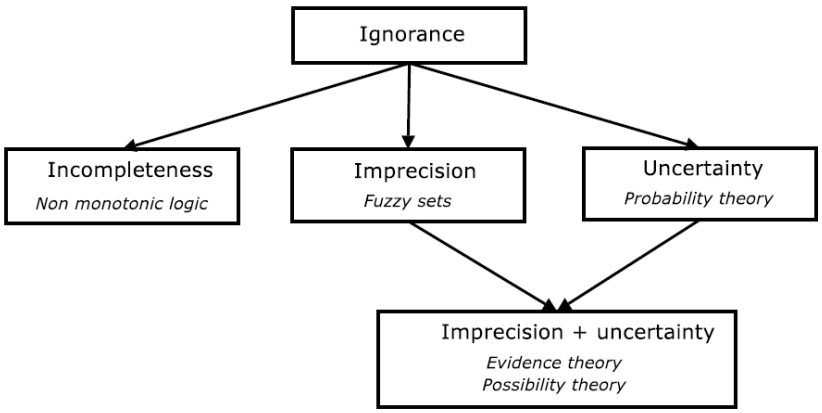

Fig. 1.1: Varieties of ignorance [1]

it has limitations when not all the *a priori* probabilities are known [1]. On the other hand, one of the proposed approaches for reasoning with imprecise and uncertain data is the Dempster-Shaffer Theory of evidence (D-S Theory). It was developed as a general framework for dealing with mathematical belief and it does not require the computation of *a priori* probabilities. It allows to combine evidence from multiple sources to arrive to a degree of belief of the world being reasoned on. Particularly, the Transferable Belief Model (TBM) is a realisation of the D-S Theory. It takes the concepts proposed by the D-S Theory and gives them a more practical approach [11].

1.1 Requirements

There is a need for semantic systems that support security forces in their investigations, particularly in the detection of riots. We aim to develop one such a system in this thesis. This system should fulfil the following requirements:

- **R1**: This system should deal with imprecise and uncertain information, as it is a key characteristic when dealing with real world information.
- **R2**: If this system is to be applied to surveillance and multimedia applications, it should be scalable and able to deal with large quantities of data. In our case it

means thousands of records

- **R3**: It should be interoperable and standards based, to be able to extract information from different sources, which would make the knowledge discovery task easier.

- **R4**: Finally, it should be effective in detecting riots from CCTV footage.

1.2 Research Objectives

- **O1**: To review the existing approaches on surveillance applications, and in particular how and if they handle imprecise and uncertain information and if they use Semantic Web technologies to achieve their tasks. This is a general objective, not directly tied with any of the requirements, but it will allow us to have a solid background on which to frame the research presented here.

- **O2**: To design and implement a Semantic Web framework to deal with imprecise and uncertain information to be used in the detection of riots in CCTV surveillance footage. Dealing with imprecise and uncertain information will allow us to fulfil **R1**, and the use of the Semantic Web Framework will allow us to fulfil **R3**.

- **O3**: To create a ground truth data set of CCTV recordings of riots to act as evaluation data for the proposed task and to benefit the larger community. This will allow us to ensure that **R3** and **R4** are met, as they depend on having data to test the framework.

- **O4**: To evaluate the scalability of the proposed framework by testing it with different sizes of inputs and configurations of systems. This will ensure that the proposed framework can be applied with real multimedia data with large inputs. This is directly related to **R2**, as this will allow us to ensure that the framework does indeed scales well with large input sizes.

- **O5**: To validate the framework using this ground truth of real riots recordings. This will serve as the usefulness evaluation of the proposed framework and is tied with **R4** as we will be able to tell how effective is our framework on detecting riots.

With these objectives in mind, several questions can be asked regarding the research presented here:

- **RQ1**: Have there been previous attempts at applying probabilistic reasoning to the semantic web and if so, which approaches do they use?
- **RQ2**: Can traditional Bayesian *a priori* probabilities be combined using the Transferable Belief Model?
- **RQ3**: Can the Transferable Belief Model be ported to the semantic web?
- **RQ4**: Can the Transferable Belief Model be applied to large inputs and can it be scaled in memory and execution time?
- **RQ5**: Can this new approach be applied to the task of riots detection? How effective is it?

1.3 Contributions of the Thesis

The main contributions of this work are:

- Regarding **RQ2**, we devise method for successfully combining traditional Bayesian probabilistic theory with the TBM. This allows us to combine *a priori* probabilities in a given domain using the TBM to reach a common knowledge of the presence of a given event based on its previous occurrence with other events.
- The design and development of a Semantic Web reasoner based on the Transferable Belief Model (TBM). This reasoner consists of an ontology and methods to perform the TBM operations on domain ontologies. This provides an answer to **RQ3**, as

we successfully port the Transferable Belief Model to the semantic web.

- The design and development of a Semantic Web reasoner based on the Transferable Belief Model (TBM). This reasoner consists of an ontology and methods to perform the TBM operations on domain ontologies. This provides an answer to **RQ3**, as we successfully port the Transferable Belief Model to the semantic web.

- We answer **RQ5** with an evaluation of the proposed framework using the detection of riots on CCTV footage to study its viability, and we conclude that the system effectively detects riots but there is still room for improvement.

- For **RQ4**, we implement parallel heuristics for the Transferable Belief Model and thus conclude that it can be scaled to work with large datasets.

- The creation of a freely available data set consisting of ground truth data for the presence of riot and other related concepts on more than 1,500,000 frames from CCTV recordings from the London riots of 2011.

1.4 Structure of the Thesis

The work presented in the thesis is organised as follows:

Chapter 2 presents relevant concepts for this work. It starts with a brief introduction to the Semantic Web and previous efforts at applying it to surveillance applications. A primer on The D-S theory and its TBM realisation is then presented, focusing on the motivations for its creation, and the different concepts and operations that comprise it. Finally, previous implementations of the TBM are presented.

Chapter 3 brings the first contribution of this work, in which we combine traditional Bayesian probabilities with the TBM. The motivation being that although *a priori* prob-

abilities are not needed for the TBM, if they are available, it would be desirable to be able to use them together with the TBM. We present examples and a use case for riot detection.

Chapter 4 presents the second contribution: a Semantic Web reasoner which uses the TBM to reason using incomplete and/or imprecise data. First, the TBM ontology is presented. Then, the implementation of the reasoner's operation is discussed as well as parallel heuristics for the implementations of the algorithms.

Chapter 5 evaluates and validates the contributions presented in this work in the context of detecting riots on CCTV recordings. The framework is evaluated in its performance and scalability, as well as in its ability to detect riots. Finally, the dataset used, which was created for this task and is also the final contribution of this work, is discussed.

Chapter 6 presents the conclusions, final remarks, and future work. This includes possible optimisations and improvements using big data processing technologies.

The remaining sections provides details of publications, and references. Supplementary material is provided in the appendices, including an example Java application using the TBM Framework and some detailed description of the data used for the ground truth annotations for the experiments.

Chapter 2

Background

2.1 Semantic Web

Initially, the World Wide Web was created by Tim Bernes-Lee as a way to organise information for its easy retrieval by people interested in getting to it. It comprises documents and resources interlinked via hypertext links. It has largely succeeded as evidenced by the billions of users that access tens of billions of documents everyday. Looking to expand on this, Tim Bernes-Lee proposed in [14] a web of data that can be processed by machines. He called this extension The Semantic Web. This would allow for websites to exchange information between them so the web would become a big indexed and queryable database.

The Semantic Web is comprised of a series of standards which allow the representation and exchange of data between different websites. The main concept behind it the representation of the knowledge as graphs, where the nodes are represented by collections of triples. A triple is a statement in the form Subject-Predicate-Object which allows to represent concepts and the relationship between them. The subject and the object would be two nodes in the graph and the predicate the edge that connects them. For example, in the statement "London is a city", the subject is "London", the predicate is "is a",

and the object is “city”. This simple statement would generate the graph presented in Figure 2.1. We could have another tuple linking “London” with some other knowledge. For example “Sadiq Khan is the mayor of London”, and thus we create a larger *web* of information. Using triples, knowledge can be encoded in the knowledge framework. Other important concepts and standards are:

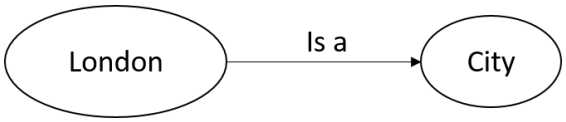

Fig. 2.1: Example of a simple graph

- **RDF/RDFs** RDF (Resource Description Framework) is the foundation of the Semantic Web. It is the most basic framework which links all the other languages and specifications [15]. It is an abstract syntax that represents the most basic relationships between concepts, effectively being the syntax on which all triples are constructed. The elements on the subject-predicate-object triples may be IRIs, blank nodes, or datatyped literals. IRIs are the standard form to identify a concept or entity. Everything except literals (constant values) are represented by IRIs. In the previous example of London, the statement would then be something like:

  <`http://example.org/London`>
  (subject)

  <`http://www.w3.org/1999/02/22-rdf-syntax-ns#type`>
  (predicate)

  <`http://schema.org/City`>.
  (object)

  Blank nodes (or B-nodes for short) are nodes that are not literals, but do not need to be identified by an IRI.

  RDFs (Resource Description Framework schema) builds on RDF and adds more expressibility. Where RDF simply relates concepts between each other, RDFs adds

additional constructs to express hierarchy between them, the most common one being `rdfs:Resource`, which models a resource class. Reasoning is the act of discovering new knowledge in the knowledge base. Since RDFs already provides a simple hierarchy between concepts, it already provides some simple forms of reasoning. For example, type inheritance can be inferred through the `rdfs:subClassOf` predicate: if in the knowledge base we have that `Bob rdf:type Human` and `Human rdfs:subClassOf Mammal` then we can infer that `Bob rdf:type Mammal`.

- **Ontology** is, in Information Technology, the set of knowledge (in the form of types, hierarchies, properties, entities, relationships) about a particular domain. An ontology can be split in two parts. The first is the TBox (from Terminological Box) which is the set of statements that encode the taxonomy of the ontology. That is, the classes, properties that define the domain itself. The second is the ABox (from Assertion Box) and it is the set of statements that define a particular instance of the ontology. For example, in a biology ontology, the statement "Cats are Animals" belongs to the TBox, whereas the statement "Garfield is a Cat" belongs to the ABox, as it contains assertions about particular instances of the concepts.

- **OWL** (Web Ontology Language) is a language to construct ontologies on the semantic web. It also builds on RDF but adds additional constructs to describe properties and classes, like cardinality, transitivity, and more. OWL 2 is the current version, which is an extension to the original 2004 version of OWL published by the W3C. OWL, being the most expressive of the languages, allows to perform more complex reasoning tasks. For example, the `owl:inverseOf` rule indicates that two properties are inverse of each other, so if we have in the knowledge base that `isParentOf owl:inverseOf hasParent`, and we have that `Bob isParentOf Alice` then we can infer that `Alice hasParent Bob`.

  OWL 2 has different profiles with different levels of expressivity, which make the reasoning more efficient in exchange of some loss of expressive power [16]. Depend-

ing on the application, users can select the profile better suited to their requirements. The three profiles are: a) **OWL 2 EL** is well suited for applications employing ontologies that define very large numbers of classes and/or properties. It is based on the $\mathcal{EL}++$ description logic (full existential qualification). On this profile ontology consistency, class expression subsumption, and instance checking can be decided in polynomial time. b) **OWL 2 QL** is better suited for applications which have large quantities of instance data and for which query answering is the most important form of reasoning. It is based on the DL-Lite family of description logics. An interesting feature of this profile is that assertions can be stored in standard relational database systems which can be queried with standard SQL queries without changes in the data. c) **OWL 2 RL** is aimed at applications that require reasoning through rules.

- **SWRL** (Semantic Web Rule Language) although currently just a member submission (since 2004), has become an alternative for representing knowledge in the form of rules in semantic databases. The rules take the form of Horn-like rules, where there is an implication between an antecedent (the body of the rule) and a consequent (the head of the rule). An example SWRL rule in human-readable format, using the previous example of the uncle rule, would be: `hasParent(?A, ?B), hasBrother(?B, ?C)` $\rightarrow$ `hasUncle(?A, ?C)`. In this example, the body is comprised of the first two clauses (before the "$\rightarrow$" symbol, which could be read as "then") and the head would be the last one. In this case only properties are being used, but literals and built-in functions can also be used.

- **SPARQL** is a query language to extract information from RDF semantic data sources. This language allows the traversal of the knowledge graph through querying required and optional graph patterns along with their conjunctions and disjunctions. SPARQL allows this knowledge graph to be spread across diverse data sources. We will explain the anatomy of a SPARQL query using the following example `SELECT` query.

```
PREFIX ex: <http://example.com/example#>

SELECT ?city ?mayor
WHERE {
?UK  a     ex:Country  ;
ex:Name ‘‘United Kingdom’’  .
?UK ex:hasCity  ?city  .
?city ex:hasMayor ?mayor .

}
```

This query returns all the cities in the United Kingdom and their mayors. The first line of the query binds the namespace of the given ontology IRI to the prefix `ex:`. This helps us to compact the query, as otherwise we would have to write the entire IRI for every concept in that ontology. It is possible to have multiple namespace bindings on this section, or none at all.

The next line tells the query which variables to return. Variables in the query are prefixed by a question mark "`?`" and can have any name but it has to be consistent in the whole query. The second part of the query, the `WHERE`, is where the graph pattern to match is specified. The "`a`" keyword is a shorthand for the `rdf:type` predicate.

In this graph pattern, we are first matching a node in the knowledge graph which has an `rdf:type` relationship with `ex:Country` and an `ex:Name` property with value "United Kingdom". We traverse the graph further by adding the nodes which have an `ex:hasCity` with the previously matched node (in this case it is likely to be only one) and then the nodes which have an `ex:Mayor` relationship with the cities. Other ways of matching nodes is through the optional matches or

value constraints (also known as filters).

Other types of SPARQL queries include the `CONSTRUCT`, `ASK`, and `DESCRIBE`.

- **SPIN** (SPARQL Inference Notation) is an alternative to SWRL using SPARQL. It was submitted to W3C on 2011 by a group led by TopQuadrant [17]. The main idea behind it is to use SPARQL's CONSTRUCT queries to create the new inferred tuples in the rules. Other types of queries are also supported for other features. A way to define SPARQL functions is also defined, among other things.

- **Jena** is an open source framework for the Semantic Web written in Java. It provides support for storing and querying (through SPARQL) standards based semantic repositories. It provides programmatic support to work with graphs written in RDF and ontologies in OWL, among others.

2.2 Semantic Web Technologies for Surveillance Applications

Gomez-Romero *et al.* [13] developed a framework which applies logical reasoning to an OWL model of a classical visual tracker's output from surveillance videos and applies abductive reasoning to automatically discover information. They define and successfully apply a rule for detecting mirrored persons in the window of a shop in a surveillance video. Again Using semantic technologies, a layered meta-data model for video surveillance system is presented in [18]. In particular they use OWL to model the taxonomy, SWRL and SPARQL to define rules and queries, and the Pellet reasoning engine. The use of MPEG-7 visual descriptors allows a high interoperability in the information capture process. The framework is validated on very simple scenarios like detecting persons or detecting duplicated objects in the tracking phase. Another framework for visual analysis using semantic technologies is presented in [19]. Their main contributions include the dynamic reconfiguration of the framework's work-flow at runtime depending

on several factors like domain of interest, user preferences, and system capabilities (i.e. available visual analysis methods). In particular, the framework is validated on a surveillance domain, defining the ontology on Protègè and available visual analysis methods on OpenCV. At runtime, different processing capabilities are added or subtracted, and user preferences (like priority for processing time, memory consumption or accuracy) are also modified. The test cases are simple scenarios like abandoned object detection. The system successfully reconfigures itself at runtime according to the different conditions. As can be seen, the approaches presented here use non monotonic logic to handle incompleteness, but imprecision and uncertainty are not taken into account.

In our previous work in [20], we use SWRL rules to detect high level events on surveillance footage. We used mid level concepts and events extracted from computer vision algorithms. We exploit the fact that usually events follow a logical sequence of events. What we do then is encode that sequence of events in a SWRL rule and apply the Pellet reasoner to extract new tuples. For example, when there is a robbery, the sequence of events could be described as a person approaching another person, then suddenly the first person starts running when they reach the second person, and finally the second person running behind. We successfully apply this approach to detect four kinds of events: Pick Pocketing, Beat and Run Away, Vandalism Against Walls, and Vandalism Against Cars.

2.3 Introduction to the TBM

This section presents a primer on the Dempster-Shafer (D-S) Theory. We present first the motivations behind the creation of this theory. We also introduce the Transferable Belief Model, which is a realisation of the D-S Theory. We use an example to infer the likeliest perpetrator of a crime to introduce the concepts.

2.3.1 Bayesian Probability

Probability theory allows us to try and predict the outcome of random events. Consider the experiment of throwing a fair dice. There are 6 possible outcomes (or events) to this experiment, and because we are dealing with an honest dice, all of them are as likely as the others. This means that each event has a possibility of 1/6. We are going to use the same example to introduce some mathematical definitions and properties of probability:

- Ω is the set of all the possible outcomes of the experiment. It can be called the sample space, the power set, or the frame of discernment. In our example, $\Omega = \{1, 2, 3, 4, 5, 6\}$ which are all the possible values the dice can take when throwing it.

- $f(x)$ is a probability value assigned to $x \in \Omega$. Each event has a probability assigned to it.

- $f(x) \in [0, 1]$ for all $x \in \Omega$. This means the probabilities have to be a real number between 0 and 1. In our example, the probabilities for all the events are $1/6 \approx 0.16$.

- $\sum_{x \in \Omega} f(x) = 1$ all of the probabilities in the sample space have to add up to one. In our case, all the 6 probabilities are $1/6$ and $1/6 \times 6 = 1$.

- if E is an event, $P(E)$ is the probability of that event being the outcome of the experiment and it is expressed as: $P(E) = \sum_{x \in E} f(x)$. This means that the probability of that event happening is the summ of all the outcomes where that event is present. For example $P(\{2, 3\}) = 2/6$.

Bayesian probability is an interpretation of probability in which the concept of probability is interpreted as reasonable expectation of the event to measure, based on previous occurrences of the experiment. There are two views of the Bayesian probability, the objectivist view treats the probabilities as a state of knowledge of the world. The subjectivist view sees the probabilities as an assessment of the personal belief of the state of

the world. It is an extension of propositional logic where you reason with a probability assigned to a hypothesis and test it.

Bayesian Probability bases itself in Bayes' Theorem that relates the probability of one event A with the probabilities of another event B. A popular example is the relationship between cancer and age, so knowing a person's age can more accurately help us assess the probability of the person having cancer.

In particular, Bayes' Theorem states that:

$$P(A|B) = \frac{P(B|A)P(A)}{P(B)}$$

Where A and B are events, and:

- $P(A)$ and $P(B)$ are the independent probabilities of observing A and B.
- $P(A|B)$ is the probability of observing A if we observe B. $P(B|A)$ is the probability of observing B given that we observe A.

2.3.2 The Demster-Shafer Theory

In this section we will introduce the motivations behind the initial ideas by Arthur P. Dempster in [21] and further developments by Glenn Shafer in [22]. We will use the unreliable sensor paradigm introduced in [23] but adapt it to the computer vision domain to present the Dempster-Shaffer Theory and some general differences with the Bayesian probability theory.

Imagine that we have a computer vision algorithm to detect persons on images. Through testing, we know that this algorithm is accurate 80% of the time. The other 20% the results are inaccurate due to different conditions like occlusion, lighting conditions, etc. and the result is not related to the actual existence of persons in the scene.

This leaves us with a frame of discernment of three sets:

$$\overbrace{\underset{\begin{pmatrix} Pers \\ NoPers \end{pmatrix}}{State} \times \underset{\begin{pmatrix} Pos \\ Neg \end{pmatrix}}{Output} \times \underset{\begin{pmatrix} Acc \\ NotAcc \end{pmatrix}}{Accuracy}}^{\Omega(\text{Frame of Discernment})}$$

Now suppose we run the algorithm on an image and we get a positive output indicating that there is a person. What is the probability that the result is correct? In classic probability theory this means we want to know $P(Pers)$ knowing Pos: $P(Pers|Pos)$. We know that:

- $P(Acc) = 0.8$ and $P(NotAcc) = 0.2$
- $P(Pers|Pos, Acc) = P(NoPers|Neg, Acc) = 1$ since when the algorithm is accurate, the output will be the actual state of the scene
- $P(Pers|Pos, NotAcc) = P(Pers)$ since when the result is accurate, we cannot infer the actual state of the scene
- $P(Pos|Acc) = P(Pers)$ because when the result is accurate, the output corresponds to the state of the scene

If we apply the Bayes' rule to our hypothesis, and try to solve, we get:

$$\begin{aligned}
P(Pers|Pos) &= P(Pers|Pos, Acc)P(Acc|Pos) + P(Pers|Pos, NotAcc)P(NotAcc, Pos) \\
&= 1\frac{P(Pos|Acc)P(Acc)}{P(Pos)} + P(Pers)\frac{P(Pos|NotAcc)P(NotAcc)}{P(Pos)} \\
&= \frac{0.8P(Pers)+0.2P(Pers)P(Pos|NotAcc)}{P(Pos)} \\
&= \frac{P(Pers)(0.8+0.2P(Pos|NotAcc))}{P(Pos|Acc)P(Acc)+P(Pos|NotAcc)P(NotAcc)} \\
&= \frac{P(Pers)(0.8+0.2P(Pos|NotAcc))}{0.8P(Pers)+0.2P(Pos|NotAcc)}
\end{aligned}$$

Here we face a difficulty because in order to compute $P(Pers|Pos)$ we need probabilities of $P(Pers)$ (The general probability there is a person in the scene) and $P(Pos|NotAcc)$

(The probability that the result is positive when the output is not accurate). To solve this, a Bayesian could try to measure these probabilities when possible, or simply assume values for them. The quality of the conclusions will be linked to the quality of the assumptions.

A third option would be to use the Upper and Lower Probability model (ULP), where we assign superior (Π_{sup}) and inferior (Π_{inf}) boundaries to the unknown values. In our case, we know that they are probabilities. That means they can take values from 0 to 1. This would give the following inferior and superior bounds to our hypothesis:

- $\Pi_{inf}(Pers|Pos) = inf_{x\in[0,1]^2} P(Pers|Pos) = 0$
- $\Pi_{sup}(Pers|Pos) = sup_{x\in[0,1]^2} P(Pers|Pos) = 1$

We can see that this is not a feasible approach because we are still in total ignorance because of the solution space is composed of all the possible values.

Dempster proposed then to reason without Bayesians' *a prori* probabilities in the ULP model and instead work only with the known information.

In our computer vision algorithm example, we know the probability space on the variable *Accuracy*. We can map these known values to compatible events in Ω. For example, if we know that the output was accurate, this is not compatible with the event $(Pers, Pos, NotAcc)$. This gives us a multi-valued mapping (Γ) to Ω, represented in Figure 2.2.

Intuitively, the lower probability of $A \in \Omega$ can be defined as the sum of probabilities of all the events in the known probability space that completely support A (i.e. when the subset is the same subset we are looking for) and the upper probability can be defined as the sums of probabilities of subsets that completely support A plus the events that partially support A.

We now run the algorithm and the output is positive, meaning that the algorithm

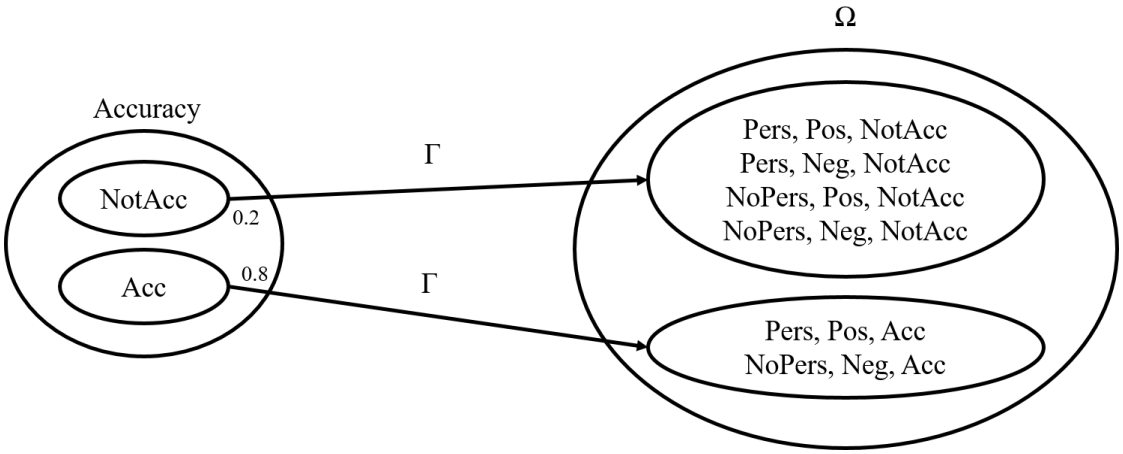

Fig. 2.2: Dempster's multi-valued mapping

thinks that there is a person present in the scene. We update our knowledge base to remove those events that are incompatible with the new knowledge. The updated mapping is presented in Figure 2.3.

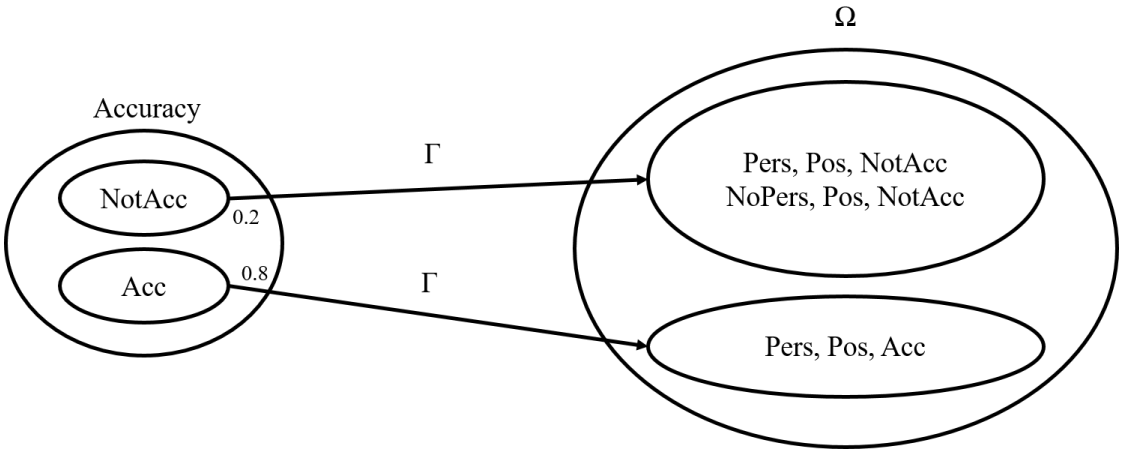

Fig. 2.3: Updated mapping when the algorithm is run and the output is positive

In our example, the lower probability that there is a person when the algorithm output is positive is 0.8. We can say this because the only set that completely supports $(Pers, Pos)$ is $(Pers, Pos, Acc)$ and has a probability of 0.8. The upper probability would be 1, because apart from the set that completely supports it (the previous one)

we have another one that supports another irrelevant set (so its support is not complete).

Dempster also proposed a way to combine information from different sources. This method will be explained later and we will see it is one of the fundamental aspects of the Dempster-Shaffer theory.

Glenn Shaffer, after assisting one of Arthur Demspter's course on statistical inference at Harvard, offered in [22] a reinterpretation of Dempster's ideas. He introduced, among other things, the belief functions. He shifted from the multi-valued mapping to mass functions.

2.3.3 The Transferable Belief Model

The TBM is one of many proposed realisations of the Dempster-Shaffer theory. It was introduced by Philippe Smets in [11]. It proposes some mathematical formalisms backed up by Dempster-Shafer's theory and belief masses and functions.

We will use a single variable example based on one presented in [1] to introduce one by one the concepts of the TBM. In our version of the example, we capture CCTV footage of one rioter (with his face covered) breaking into a store to loot it. There are three suspects of this crime: Carl, Peter or Michael. This leads us to the first concept:

Frame of discernment (Ω) As discussed before, it is the finite set which holds all the hypothesis of the task. In our example, $\Omega = \{Carl, Peter, Michael\}$. In a closed world context, the truth must be inside this frame of discernment. In an open world context, the truth may be somewhere else. For this example, we will use the closed world assumption, but it must be noted that OWL and related ontology languages usually use the open world assumption, but reasoning can be performed on a closed world knowledge.

Suppose now that we are told by an old person that he thinks he saw Carl or Peter do it, but he is not sure. This now leads us to the next two concepts: Mass function and

focal elements.

Mass function The mass (m) is a quantifiable amount of support to a group of hypothesis in Ω. This support is introduced by the Evidence. Assigning a mass $m(A)$ to a subset A of Ω, gives support to exactly that subset A. As in traditional probability theory, the mass function for particular evidence must verify that $\sum_{A\subseteq\Omega} m(A) = 1$. In our case, we subjectively select the value of 0.8 to the confidence of the testimony of the old person. This means that $m(Carl, Peter) = 0.8$. We need to include the fact that this person might be wrong, but we should be careful not to assign the rest of the mass to $Michael$, because the testimony does not support directly the fact that the perpetrator is Michael. This means that $m(Carl, Peter, Michael) = 0.2$.

Focal elements are the subsets of Ω having non-null mass. The set of focal elements is called Focal Set (FS). In our case, for this particular evidence we just learned, $FS = \{\{Carl, Peter\}, \{Carl, Peter, Michael\}\}$.

In the CCTV footage, it is evident that the perpetrator has long hair, and we know that only Peter and Michael have long hair. We must now combine the new information with the current state of the knowledge base. For this we must introduce two new concepts: Potential and Dempster's Rule of Combination.

Potential is the formal way of defining the evidence available to the system. It is the mass function induced by particular evidence. In our case, we have two potentials produced by the two testimonies we have right now.

$$p_1 = \{\{Carl, Peter\}\,[0.8], \{Carl, Peter, Michael\}\,[0.2]\}$$

$$p_2 = \{\{Peter, Michael\}\,[0.9], \{Carl\}\,[0.1]\}$$

Dempster's Rule of Combination currently we have two potentials p_1 and p_2, each with its own mass functions m_{p1} and m_{p2}. The goal is to get a single, combined potential $p_{1\oplus 2}$ with a joint mass function $m_{p1} \oplus m_{p2}$. Dempster's Rule of Combination relies on the intuition that the product $m_1(X) * m_2(Y)$ supports $X \cap Y$. This means that:

$$m_{1,2}(A) = (m_1 \oplus m_2)(A) = \sum_{B \cap C = A \neq \oslash} m_1(B) * m_2(C)$$

For the current state of our knowledge base, we get the following generated Focal Elements:

$$FE_{p1\oplus p2} = \{\{Peter\}, \{Carl\}, \{Peter, Michael\}\}$$

and the following masses:

$$m(\{Peter\}) = 0.8 * 0.9 = 0.72$$

$$m(\{Carl\}) = 0.8 * 0.1 + 0.2 * 0.1 = 0.1$$

$$m(\{Peter, Michael\}) = 0.2 * 0.9 = 0.18$$

for a combined potential of:

$$p_{1\oplus 2} = \{\{Peter\}\,[0.72], \{Carl\}\,[0.1], \{Peter, Michael\}\,[0.18]\}$$

A situation arises when we introduce new evidence that contradicts the currently available potentials. Let us now assume that from a different CCTV source, we see Carl near the scene of the crime. The newly introduced potential (with the subjectively selected values of 0.8 and 0.2 to the confidence of the information) is:

$$p_3 = \{\{Carl\}\,[0.8], \{Peter, Michael\}\,[0.2]\}$$

If we try to combine this potential into the knowledge base we will notice that $\{Peter\} \cap \{Carl\} = \oslash$ and $\{Carl\} \cap \{Peter, Michael\} = \oslash$. This means we have a conflict. This is evident when we combine the new potential and get empty focal elements and this happens because one potential is strictly supporting a set of hypothesis, and the other one supports a completely disjoint set of hypothesis. In TBM, we can quantify this contradiction (known as the **conflict** k) as the sum of the product of the masses of the conflicting focal elements:

$$k_{1,2} = \sum_{B \cap C = \oslash} m_1(B) * m_2(C)$$

In our case:

$$k = 0.72 * 0.8 + 0.1 * 0.2 + 0.18 * 0.8 = 0.74$$

Under the open world assumption, a conflict means that the truth lies outside the current frame of discernment and we currently do not posses the knowledge needed to make sense of it. In that case, the conflicting mass would go to $\oslash$ ($m(\oslash) = k$). In a closed world assumption, to keep the mass functions adding up to 1, the produced mass function must be normalised with the conflicting mass. This means that the combination becomes:

$$m_{1,2}(A) = (m_1 \oplus m_2)(A) = \frac{1}{1 - k_{1,2}} \sum_{B \cap C = A \neq \oslash} m_1(B) * m_2(C)$$

In our closed world example, this means that:

$$FE_{p1 \oplus p2 \oplus p3} = \{\{Peter\}, \{Carl\}, \{Peter, Michael\}\}$$

$$m(\{Peter\}) = \frac{0.72 * 0.2}{1 - 0.74} \approx 0.55$$

$$m(\{Carl\}) = \frac{0.1 * 0.8}{1 - 0.74} \approx 0.3$$

$$m(\{Peter, Michael\}) = \frac{0.2 * 0.18}{1 - 0.74} \approx 0.14$$

$$p_{1 \oplus 2 \oplus 3} = \{\{Peter\}\,[0.55], \{Carl\}\,[0.3], \{Peter, Michael\}\,[0.14]\}$$

The goal of the task is to find the perpetrator of the crime. We will now introduce some functions that will allow us to quantify our knowledge of the system according to the available evidence.

Belief is the justified amount of support given to any subset of Ω. Any mass that includes a given subset, supports that subset. This is Shafer's interpretation of the lower probability. Mathematically:

$$bel(A) = \sum_{B|B \subseteq A} m(B)$$

For this exercise we are interested in knowing our belief, according to the evidence, that each of the elements in the frame of discernment is the perpetrator:

$$bel(\{Carl\}) = 0.3$$

$$bel(\{Peter\}) = 0.55$$

$$bel(\{Michael\}) = 0$$

Plausibility is the total amount of support given to A at least partially. More specifically, is the support not given strictly to $\overline{A}$. This is Shafer's interpretation of the upper probability.

$$pls(A) = 1 - bel(\overline{A}) = \sum_{B|B\cap A\neq\oslash} m(B)$$

In our example:

$$pls(\{Carl\}) = 0.3$$

$$pls(\{Peter\}) = 0.55 + 0.14 = 0.69$$

$$pls(\{Michael\}) = 0.14$$

It can be seen intuitively that $bel(A) \leq pls(A)$.

Ignorance is the difference between $pls(A)$ and $bel(A)$. It is the support that is only given partially to A. Intuitively we can see that these are masses allocated not exclusively to A, which means that we do not have enough information to discriminate between the support to A and all the other members of the focal elements where A is present.

$$ign(A) = pls(A) - bel(A)$$

Doubt is the degree of support that will never be assigned to A.

$$dou(A) = 1 - pls(A) = bel(\overline{A})$$

So far, what has been done is simply asserting beliefs. But the goal of all of this is

to make a decision. Smets makes the distinction of two steps in the TBM:

- **Credal** level (from the Latin *Credo* "To Believe") is where beliefs are assessed, updated, and combined.
- **Pignistic** level (from the Latin *Pignus* "Bet") is where decisions must be made based on the beliefs.

Once the knowledge base is updated with all the available information, all the potentials are combined, and all the beliefs are updated, we must select from Ω the best hypothesis or group of hypothesis. There is no standard rule for this, and usually rules are created for specific problems, commonly involving the belief and plausibility functions mentioned before. For example, selecting the maximum of belief or plausibility, or a combination of both. More advanced rules are available but will not be discussed here.

In our example, we can see that the perpetrator is most likely Peter, as is not only the hypothesis with the highest amount of support (highest belief), it is also the most plausible.

One of the strengths of the TBM is its ability to combine knowledge from many heterogeneous sources of information, with knowledge on different variables.

When the problem's domain lays in multiple variables, there are certain steps that must be taken in order to perform the reasoning, for example if the evidence you get only pertains to some variables and not all. Or if you want to measure the belief only in some of the variables. This is very useful because it allows us to assess only the interesting parts of the problem, and we can introduce new knowledge to the system even if this knowledge works with variables that we do not presently include. In our example of the looting rioter, we might also be interested for example in the weapon he used to break in (if there is no footage available), or in his accomplice. We are able to reason on all the variables by separate or in any combination of them.

The combination of two potentials from different domains can be performed thanks

to two operations: Extension and Marginalisation.

Extension is the operation of introducing new variables in a potential. In mathematical terms, if we have a potential ϕ on a domain D, its extension on $D2 \supset D$ consists of applying a cylindrical extension of the focal elements of ϕ. For example let us suppose

- we have Ω with two variables X and Y with instances $\{x1, x2\}$ and $\{y1, y2\}$ respectively
- we have potential ϕ defined on $D = X$ with masses $m(\{x1\}) = 0.2$, $m(\{x2\}) = 0.8$
- the extension of ϕ on $D2 = X, Y$, noted $\phi^{\uparrow D2}$ is:
  - $m(\{(x1, y1)\} \cup \{(x1, y2)\}) = 0.2$
  - $m(\{(x2, y1)\} \cup \{(x2, y2)\}) = 0.8$

Marginalisation is the operation of removing variables from potentials. Mathematically, if we have potential ϕ on domain D and we want to marginalise it on $D2 \subset D$, we must project its focal elements on $D2$ and regrouping any identical resulting focal elements.

For example, using the same frame of discernment Ω from the previous example

- we want to marginalise ϕ which is defined on $D = X, Y$ and has masses:
  - $m(\{(x1, y1)\}) = 0.1$
  - $m(\{(x1, y2)\}) = 0.1$
  - $m(\{(x2, y1)\}) = 0.8$
- the marginalisation of ϕ on $D2 = X$, noted $\phi^{\downarrow D2}$ is defined as:
  - $m(\{x1\}) = 0.1 + 0.1 = 0.2$

– $m(\{x2\}) = 0.8$

Combination of multi-variable potentials is achieved by extending the two potentials by the combined domains and applying the standard combination. Mathematically, if we have potentials ϕ_1 and ϕ_2 with domains $D1$ and $D2$ respectively, the operation we must perform is $\phi_{12} = \phi_1^{\uparrow D1 \cup D2} \oplus \phi_2^{\uparrow D1 \cup D2}$ The resulting potential ϕ_{12} will be defined in the domain $D1 \cup D2$. But it can be restricted to another domain $D3$ by simply marginalising it on $D3$. This operation is called $fusion$.

2.4 Previous Implementations of the TBM

Previous implementations of the TBM could be split in two categories. One is comprised of general-purpose frameworks which do not target a particular application. In the other group we have *ad-hoc* frameworks that solve particular problems.

Starting with the general approaches is one of the early modern attempts of implementing the TBM, presented by Haenni & Lehmann [24] in 2001. They first discuss theoretically different possibilities for implementing the DS belief functions. In particular, the encoding of the mass functions and implementations of the TBM operations, for which they also propose new efficient algorithms. Their solution is based on the binary representation of the focal sets, and as programming language used MCL 4.3 (Macintosh Common Lisp), and all their tests were performed on a 400-MHz Power Mac G3 with 768 MByte of RAM.

The experiments consisted in the marginalisation of the combination of two belief potentials. In total, 24 implementation variants are tested, split into three categories: classical methods (the two potentials are combined, and then marginalised in the new domain); Stepwise marginalisation (similar to the previous one, but the marginalisation is performed on a step-by-step procedure); and using the fusion operation, proposed by

them as a more efficient alternative. The test bed consisted of two random potentials with 632 binary variables and 1101 initial belief potentials. As conclusions, they confirm the computational complexity of the TBM, but manage to reduce the execution time of the most expensive operation in the framework. They reach three implementations alternatives which yield better results than previous approaches.

Burrus & Lesage [1] describe in their technical report an implementation of the TBM made in C++ called `eVidenZ`. Their implementation is largely based on the one by Haenni & Lehmann [24]. They also propose a new method for building the knowledge system once and assigning the mass values later. They called this method "Delayed Mass Valuation". They illustrate the new method with a hypothetical medical diagnosis application which uses the TBM. The patient is diagnosed with a series of questions about the symptoms:

- "Do you have headache?"
- "Do you have fever?"
- ...(any other question that might help with the diagnosis)...

Each patient would give the answers relevant to his case, but this means that every time a new patient is interrogated, the knowledge base has to be rebuilt. They propose to build the knowledge base once, and belief masses are provided by different contexts (i.e. each patient is a new context).

They do manage to achieve better time performance, thanks to a combination of factors that include the choice of programming language, the use of newer hardware, and efficient implementation of the algorithms.

Their tests consisted on measuring the computation time and memory consumption of the combination of two randomly generated potentials. To test how different characteristics of the potentials affect the performance of the framework, they performed different tests varying different parameters of the focal elements. Namely:

- Number of variables
  - 2 to 12 variables
  - 2 realisations per variable
  - 1000 focal elements per potential
  - 100 configurations per focal element
- Number of realisations per variable
  - 2 variables
  - 500 to 4000 realisations per variable
  - 100 focal elements per potential
  - 100 configurations per focal element
- Number of focal elements per potential
  - 2 variables
  - 500 realisations per variable
  - 100 to 700 focal elements per potential
  - 100 configurations per focal element
- Number of configurations per focal element
  - 2 variables
  - 1000 realisations per variable
  - 200 focal elements per potential
  - 100 to 250 configurations per focal element

One drawback of their approach is that they also use bit representation of the focal elements, the storage requirements grow exponentially with each variable. But they claim it might not be critical as problems usually do not involve that many variables [24]. Empirical tests on the Delayed Mass Valuation approach show its infeasibility, but they identify a few aspects where it can be improved. More recently but on the same research lab, their work was later ported to Java with the evidence4j framework [25]. There is no evaluation published as of yet on this framework.

Moving to the more specific frameworks, Klein *et al* [26] and Munoz-Salinas *et al* [27] use the TBM together with Particle filters similarly for computer vision problems. They both take "evidence" from different "sensors" (in this case the sensors being computer vision algorithms and the evidence their output) and combine it using the TBM to arrive to a knowledge which takes into account all the evidence. They adjust the reliability of the sensors depending on several factors, for example, the opinions on occluded objects are less reliable than ones that are fully visible. Both approaches achieve good results, but are otherwise not very specific on implementation details. First, Klein *et al* [26] use it to detect the cars that are in front with footage taken from a moving car. Their evidence comes from:

- Colour distribution for the shadow beneath the car
- Colour co-occurrence matrix based method for the car's colour-texture information
- Symmetric cards drawn from an image of contours are used for the car's shape
- A computer vision tracking algorithm is used for the speed

Their frame of discernment (Ω) is comprised of three variables. For the proceeding car that is being tracked, the frame is split into subwindows. And for each subwindow, the hypothesis are:

ω_1: Contains a car

ω_2: Contains background

ω_3: Contains another car

A Potential which contains beliefs on subsets of Ω is created from the output of each algorithm, and then all the Potentials are combined to create a "global" potential which contains the knowledge of all the sources of evidence.

Then Munoz-Salinas *et al* [27] have a similar approach to track people in a multi-camera setting. For the "evidence", they start with multiple calibrated cameras that share a common reference system. A person tracking algorithm is employed on each of the frames to extract the presence of persons in a given shot. Each tracker also has shape information and a colour model for each of the objects being tracked. The variables for their domain is the presence or absence of the person being tracked in the scene. The belief masses are calculated from the output of the algorithm and weather a particular instance can be trusted or not due to unfavourable conditions (i.e. occlusion of the person). The knowledge of all the cameras is then combined using Dempster's Rule of Combination.

2.5 Uncertainty and Imprecision in the Semantic Web

There are already some efforts for encoding probabilistic information in the Semantic Web using OWL and/or RDF. These efforts usually handle some other form of uncertainty or offer a general way to encode any probabilistic information in the Semantic Web. In [28] for example, an approach is proposed for modelling and reasoning with Bayesian networks for the task of ontology mapping. [29] also focuses on Bayesian Networks, but not just for ontologies mapping. It proposes a vocabulary to model Multi-Entity Bayesian Networks. The actual task of reasoning is left for specific tools. In [30] both a model and a probabilistic reasoning engine using Markov Logic is proposed. The W3C has also started evaluating the standardisation of probabilistic ontologies, as can be seen

in the efforts expressed in the W3C Incubator Group on *Uncertainty Reasoning for the World Wide Web* [9] [31]. Additional proof of the growing popularity of this field can be seen in the yearly International Workshop on Uncertainty Reasoning for the Semantic Web (URSW), currently on its 12th edition [32]. A review of some of these efforts and some others can be seen in [33].

In the forensic and surveillance domain, Han *et al.* attempted first to use subjective logic to handle uncertainty and subjective logic to handle incompleteness using Semantic Web technologies. In [34] several forensic questions are tried to answer like 'who is the suspect of the event?' and 'who is the most probable witness of the suspect of the event?'. Data is modelled in OWL and in this first approach, it is annotated manually. This framework is then successfully tested on the detection of simple events like two people speaking.

Han *et al.* further explore the feasibility of subjective logic for surveillance and forensic scenarios in [12] and default reasoning is considered for dealing with incompleteness. Here, appropriate operators for dealing with surveillance data using subjective logic and default reasoning (but not specific to Semantic Web technologies) are introduced. Successful examples are presented for identity inference and theft inference, with and without contextual cues. Details about the implementation, if any, are not reported.

This approach is extended in [35] for estimating whether one person could serve as a witness of another person in a public area scene. To deal with the uncertainty, a reputational subjective opinion function for the spatial-temporal relations is developed. In addition, the acquired opinions are accumulated over time using subjective operators. A preliminary test case is performed on an airport surveillance, manually annotated, one minute video. Logic Programming with the CLIPS rule engine [36] is used. Large scale and more complex scenarios tests are still missing.

Others have attempted to include ignorance handling (particularly Imprecision and Uncertainty) extensions to web semantic languages (albeit not specific to surveillance or

forensic scenarios). In [37], Ceolin *et al.* proposed three extensions and applications of subjective logic in the Semantic Web, namely: the use of semantic similarity measures for weighing subjective opinions, a way for accounting for partial observations, and the new concept of 'open world opinion', i.e. subjective opinions based on Dirichlet Processes, which extend multinomial opinions.

Subjective Logic is also proposed in [38] as a way to handle uncertainty on the Semanatic Web. They call their approach Subjective DL-Lite (SDL-Lite), and it is an extension of DL-Lite with Subjective Logic. On their approach, they extend every assertion with an opinion. This can later be used to query and reason on the knowledge base. [39] is an application of Subjective Logic on Description Logics. In this case, they use it to merge knowledge from different datasources taking into account their *trustworthiness*. They do this in fours simple steps: a) information is merged in the database, b) conflicts are detected, c) compute the trustworthiness of each assertion based on its datasource, and finally d) facts with the lower trustworthiness are removed from the knowledge base to remove the conflicts. Evaluation on real life scenarios and evaluation is left for future work.

Fuzzy logic is a popular approach for dealing with imprecise information on the semantic web. [40] proposes fuzzy ontologies using OWL 2 annotation properties along with constructs particular to fuzzy logic, which would allow to handle imprecise data on a regular semantic framework. Stoilos *et al.* also propose in [41] theoretical fuzzy extensions for OWL. Their approach is later complemented in [42], [43], [44], [45], [46], and [47]. Their approach is called f-OWL. They present the abstract syntax and semantics for it. They also propose some extensions to f-OWL to handle for example fuzzy one-of relationships and XML serialisation. a fuzzy Tableaux is also introduced. To query a knowledge based under this very same paradigm, fuzzy extensions for SPARQL are proposed first in [48] and later extended in [49]. They propose the use of specially formatted comments in regular SPARQL queries to specify thresholds and other fuzzy values in the query. This allows them to maintain backwards compatibility with the

existing standard. We will show the anatomy of a fuzzy SPARQL query with an example used in this document. For example, this is a query for a casting agency that is looking at a particular type of model:

```
#TQ#
SELECT ?x WHERE {
?x rdf:type Model . #TH# 1.0
?x rdf:type Tall . #TH# 0.7
?x rdf:type Light . #TH# 0.8
}
```

This query returns models that are tall and slim. In this query, the top `TQ` is a prefix used to indicate the type of query. In this case, it is a Threshold Query, but another option is `GFCQ:SEM=XXXXX` to declare a general fuzzy query, with fuzzy threshold semantic functions, where "`XXXXX`" is the semantic function. The `TH` on each line in the where indicate the actual threshold on each statement, but in a general fuzzy query, `DG` can be used to specify the degree. Their implementation is based on the ONTOSEARCH2 system, but with extensions to support the fuzzy queries. Optimisations for scalability are also presents, with the evaluation showing positive results.

The same team also proposes fuzzy extensions for RuleML [50] and SWRL [51] (which uses RuleML XML syntax). In their work, an example of an f-SWRL rule to represent that "one is Thin if one is Tall (with importance factor 0.7) and Light (with importance factor 0.8)" could be as follows:

```
Tall(?p)*0.7  Light(?p)*0.8 → Thin(?p)
```

2.6 Applications of the TBM in Semantic Web

Existing applications of the TBM in Semantic Web fall largely in the specific or *ad-hoc* frameworks. Nikolov *et al* proposes in [52] a theoretical framework for solving the prob-

lem of ontology matching using the DS theory. Their motivation for such a framework is that automatic information extraction from textual information in the web is not 100% reliable. This might produce inconsistent information on the knowledge base if you consider the different methods of extraction and the different sources of information. To produce a valid knowledge base, this heterogeneous (and possibly contradictory) information must be combined in a reliable way. Their approach consist of four steps, starting from an inconsistent ontology:

1. **Inconsistency Detection** A subontology of all the axioms contributing to an inconsistency is selected.

2. **Construction of Belief Network** The subontology of the previous step is translated into a belief network.

3. **Assigning Mass Distributions** Mass distribution functions are assigned to nodes.

4. **Belief propagation** Uncertainty is propagated through the network and confidence degrees of ABox statements are updated.

They illustrate and theoretically validate the framework with a hypothetical finance application. However, apart from stating that the inconsistent axioms are extracted using the Pellet reasoner, there are no details on the implementation.

Rizzo *et al* have been using the TBM in several of their works together with the Semantic Web. In [53] and [54] they try to infer approximate assertions in the ABox using the TBM. They infer three types of relations: class membership, data-type fillers, and relationships between individuals. Their approach consists of first selecting the most likely candidates using the Nearest Neighbour algorithm. Then using a combination rule derived from Dempster's rule of combination, the evidence provided by the nearest neighbours is fused together to create new knowledge in the knowledge base. An interesting feature of their approach is that whenever there is conflict after the combination of the evidence, they do not normalise the mass of the resulting Focal Elements, instead they

assign that mass to a special *conflict* node. This allows to check the consistency of the ontology. They test their approach in several publicly available ontologies with positive results. Later in [55], they try to achieve a similar goal, but combining terminological decision trees with the Dempster-Shaffer Theory, getting again similarly good results.

The work done by Bellenger *et al* in [2] is probably the most similar to the TBM reasoner presented in this document, but with some fundamental differences. They first propose an ontology which models the Dempster-Shaffer Theory in OWL2. The ontology is presented in Figure 2.4

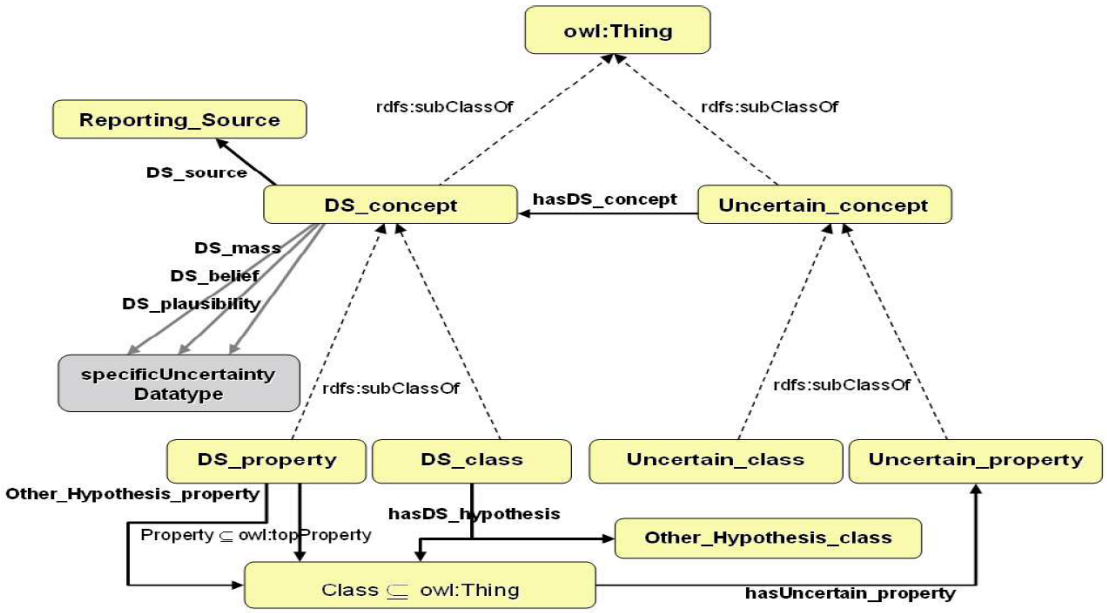

Fig. 2.4: DS-Ontology [2]

For their approach, domain ontologies use the concepts from the DS-Ontology in a given way to express uncertainty. The main difference with the approach presented in this document is that the Potentials are not created directly. Instead, the ontology and the reasoner are designed in such a way that the hypothesis are modelled as a previous step to a process which converts them into the actual potentials using a semantic similarity measure. The hypothesis are converted into sub-hypothesis which consist of intersections of all the available hypothesis, and the combination is applied to the

generated hypothesis. This is better explained with an example. Imagine a domain where the task is to identify a far away object, and you have two sources of information: a human and a radar. The ontology for this domain is presented in Figure 2.5.

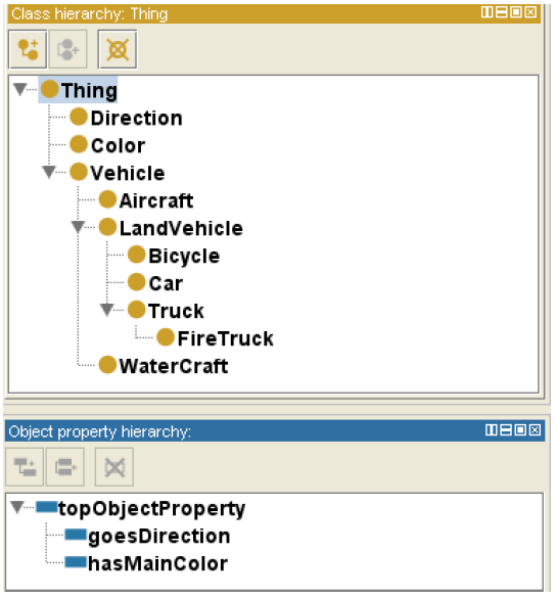

Fig. 2.5: Example object tracking ontology [2]

The radar detects a far away object, and since it is only good at differentiating between land and air objects, it assigns the following beliefs: $m(\{\texttt{:landVehicle}\}) = 0.6$; $m(\{\texttt{:aircraft}\}) = 0.1$; $m(\{\texttt{:landVehicle, :aircraft}\}) = 0.3$. The human assigns the following subjective belief masses: $m(\{\texttt{:car}\}) = 0.2$; $m(\{\texttt{:fireTruck}\}) = 0.4$; $m(\{\texttt{:landVehicle}\}) = 0.4$. The hypothesis are then encoded in the ontology as presented in Figure 2.6.

Concepts with any sort of intersection are decomposed and their intersecting parts are instead used. The generated hypothesis are as such:

- `:aircraft` $= \{H_1\}$. `:aircraft` does not intersect with other concepts in the ontology, so it is not decomposed.
- `:car` $= \{H_2, H_3\}$. `:car` intersects with `:firetruck` (since they are both land vehicles) but has a part that does not intersect.

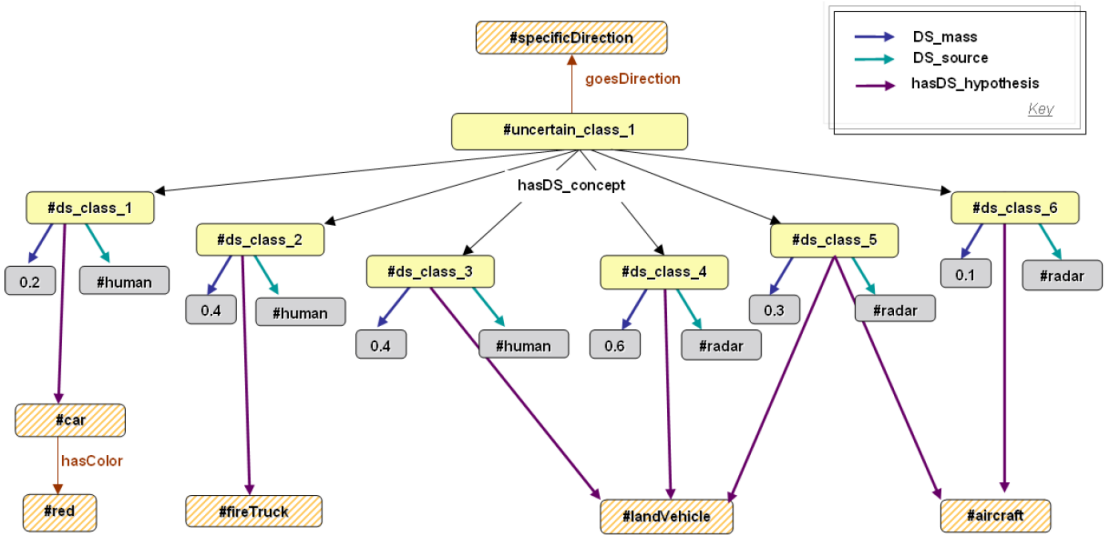

Fig. 2.6: Encoded domain using the DS-Ontology [2]

- `:firetruck` $= \{H_3, H_4\}$. `:firetruck` intersects with `:car` (as before) but has a part that does not intersect.

- `:landVehicle` $= \{H_2, H_3, H_4, H_5\}$. `:landVehicle` intersects with both `:car` and `:firetruck` but has a part that does not intersect.

The original Potentials the turn into:

- **Radar:** $m(\{H_2, H_3, H_4, H_5\}) = 0.6$; $m(\{H_1\}) = 0.1$; $m(\{H_1, H_2, H_3, H_4, H_5\}) = 0.3$.

- **Human:** $m(\{H_2, H_3\}) = 0.2$; $m(\{H_3, H_4\}) = 0.4$; $m(\{H_2, H_3, H_4, H_5\}) = 0.4$.

It is on these potentials that the combination is applied. On this approach, there are no details on the implementation or an evaluation of the framework.

Finally, Sensoy *et al* propose in [56] the use of the Dempster-Shaffer Theory to calculate the *trustworthiness* of different sources of information. Using the DS framework offers them the possibility to detect conflicts and measure the trustworthiness of uncertain data sources. This trust can be updated when new information becomes available.

They propose some methods to remove the conflict from the knowledge base:

- **Naive Deleting (ND)**: All conflicting opinions are deleted.
- **Trust-based Deleting (TDL)**: When two opinions are in conflict, the one with the lowest trustworthiness is deleted. This creates the issue that a lot of information is discarded.
- **Trust-based Discounting (TDC)**: If two opinions are in conflict, they are discounted in proportion to the trustworthiness of their sources. This approach neglects the amount of evidence used to calculate trust in sources
- **Evidence-based discounting (EDC)**: A heuristic is developed based on previous evidence to estimate the best discounting factor.

Their results show promising results for the EDC, as the system behaves as expected, discounting successfully conflicting information in highly untrusty scenarios.

2.7 Concluding Summary

We have presented in this chapter, the concepts and previous work which forms the foundation of this research. We see that the research of semantically enriched surveillance frameworks and probabilistic reasoning on the semantic web are active fields of research. However, we also see the need of a Transferable Belief Model framework for the Semantic Web, as the previous approaches presented on this chapter do not always fulfil the requirements presented in Chapter 1, as most of them focus on a particular aspect of the problem, but not all of it. As a concluding summary, we now present in Table 2-A the previous approaches and if they fulfil the requirements.

We can see that the main issue is that scalability is not considered in any of the approaches. On the following Chapters, we will address these issues by introducing a Semantic TBM Framework which we apply to the detection of riots.

Reference	R1 (Imprecise and Uncertain inf.)	R2 (Scalable)	R3 (Standards Based)	R4 (Riots/Surveillance)
2.2 Semantic Web Technologies for Surveillance Applications				
[13]	×	×	✓	✓
[18]	×	×	✓	✓
[19]	×	×	✓	✓
[20]	×	×	✓	✓
2.4 Previous Implementations of the TBM				
[24]	✓	×	×	×
[1]	✓	×	×	×
[26][27]	✓	×	×	✓
2.5 Uncertainty and Imprecision in the Semantic Web				
[28]	✓	×	✓	×
[29]	✓	×	✓	×
[30]	✓	×	✓	×
[34][12]	✓	×	✓	✓
[35]	✓	×	✓	✓
[37]	✓	×	✓	×
[38]	✓	×	✓	×
[39]	✓	×	✓	×
[40]	✓	×	✓	×
[41][42][43][44][45][46][47]	✓	×	✓	×
2.6 Applications of the TBM in Semantic Web				
[52]	✓	×	✓	×
[53][54][55]	✓	×	✓	×
[2]	✓	×	✓	✓
[56]	✓	×	✓	×

Table 2-A: Previous approaches

Chapter 3

Combining *a priori* probabilities with the TBM

In this chapter, one of the contributions of the thesis will be presented, namely the use of traditional Bayesian *a priori* probabilities with the TBM. As discussed previously, traditional Bayesian probability requires the *a priori* probability distribution taking into account the available information, and when new information becomes available, update the probability distribution using Bayes' formula. On the other hand, the TBM does not require *a priori* probabilities to be computed.

This chapter deals directly with Requirement **R1**, as we propose a new method of combining *a priori* probabilities with the TBM. We also deal indirectly with **R4** as we use the example of riot detection to showcase our approach. **RQ2** is also answered, as we prove that you can successfully combine *a priori* probabilities with the TBM. Our approach works by encoding de prior knowledge of the world using TBM potentials. This potentials have to be carefully constructed so they can encode correctly our knowledge about the world. Once the potentials are constructed, they can be combined and queried. This also allows us to combine different *a priori* probabilities even if they are calculated on different variables (different domains). In the next sections we will explore this

approach in more detail.

3.1 Motivation

In Bayes' subjectivist view of Probability, it is possible to measure the frequency of non-random events and reach a degree of *belief* that a certain event happens given that we know that some other related event happened. For example, in surveillance, if the presence of rioting is related to the presence of crowds, using Bayes' Theorem, the actual presence of crowds in a scene (the prior knowledge) can be used to assess the probability that there is also riots in the scene. This means that there are concepts that entail other concepts. Bayes' theorem states that given two events A and B, the probability $P(A|B)$ of observing A given that B is observed is:

$$P(A|B) = \frac{P(B|A)P(A)}{P(B)}$$

Where:

- $P(A)$ and $P(B)$ are the probabilities of observing A and B independently and,
- $P(B|A)$ is the probability of observing B given that we observe A

There are, however, composite concepts in which it is the presence of multiple other concepts that entails them. Take for example the high level concept "Vandalism", defined as the act of deliberate damage of property. If a certain scene contains window glasses breaking, we might not learn much else from the scene. Or we might neither if we learn by separate of the presence of fire or people with the faces covered. But if we have a scene where windows are breaking and we have the presence of fire and people covering their faces all happening at the same time, it is highly likely that there is in fact an act of vandalism occurring.

In this case, if we compute only the *a priori* probabilities of the presence of window glasses breaking and vandalism we might not learn much about the scene, and if we compute the *a priori* probabilities of all the combined concepts, we would have to do it for all the possible combinations of concepts, as for example the presence of window glasses breaking might entail also other concepts.

In the traditional Bayesian world, we would have to compute the *a priori* probabilities of "Glass Breaking" and "Vandalism", but we would also have to compute the probabilities of "Glass Breaking", "Fire" and "Vandalism", and the same for all the possible combinations of concepts. This requires going back to the "world" and measuring them, but this might be too complex or might not be available at all.

Imagine we have a CCTV scene being monitored by different kinds of sensors and computer vision trackers and detectors. In a particular instance of time, the "Glass Breaking" detector, tells us that a window has been broken. We not only include this new knowledge in the system, but we also include the knowledge of whatever concepts it entails. For example, in this example, let us assume that "Glass Breaking" entails "Vandalism" with a probability of 10% (meaning that 10% of the previous cases of a glass breaking was due to "Vandalism"), but it also entails "Earth Quake" with a probability of 3%, "Explosion" with a probability of 10%, and "Accident" with a probability of 50%. But then the Fire Detector also triggers an alarm that there is a fire. And the presence of Fire entails "Vandalism" with a 30% probability, "Accident" with a 50% probability, and "Explosion" with a probability of 10%. Please note that these are made up values to illustrate the approach with an example. In a real application, these values would be calculated by examining the data.

At this point, we can intuitively say that there is an "Accident", as it is the most likely concept both "Glass Breaking" and "Fire" entail. There are also combined probabilities for the other concepts, but with smaller values.

Calculating this using the Bayes' Rule is not an easy task. And as we saw on Chap-

ter 2, it might even lead to inconclusive results. In this particular example, if we want to compute the probability of there being an "Accident" given that there's "Glass Braking" and "Fire", we start by listing our current knowledge of the world, which are the *a priori* hypothetical probabilities presented before:

- $P(Vand|GlassBr) = 0.1$
- $P(Quake|GlassBr) = 0.03$
- $P(Expl|GlassBr) = 0.1$
- $P(Acc|GlassBr) = 0.5$
- $P(Vand|Fire) = 0.3$
- $P(Acc|Fire) = 0.5$
- $P(Expl|Fire) = 0.1$

In this particular scene we learn the presence of "Glass breaking" and "Fire", and we want to know what is the likelihood of there being an accident too:

$$P(Acc|GlassBr, Fire)$$

If we apply Bayes' Rule:

$$P(Acc|GlassBr, Fire) = \frac{P(GlassBr, Fire|Acc)P(Acc)}{P(GlassBr, Fire)}$$

Here we find that we need other *a priori* probabilities to be able to compute this. We must go back and learn the probability that there is "Glass Breaking" and "Fire" given that there is "Accident" ($P(GlassBr, Fire|Acc)$), and the independent probabilities of "Accident" ($P(Acc)$) as well as the combined probabilities of "Glass Breaking" and "Fire"($P(GlassBr, Fire)$). And this is a system with only 3 variables.

Things get more difficult as we include more variables. Let us assume for example

that we calculate the missing probabilities we need to calculate $P(Acc|GlassBr, Fire)$. But then at a later stage we learn of the presence of another event, "Screams" which hypothetically entails "Accident" with a probability of 60%. We now want to learn the probability of there being an accident, given that we have glass breaking, fire, and screams:

$$P(Acc|GlassBr, Fire, Screams)$$

Applying Bayes' Rule, we get:

$$P(Acc|GlassBr, Fire, Screams) = \frac{P(GlassBr, Fire, Screams|Acc)P(Acc)}{P(GlassBr, Fire, Screams)}$$

The *a priori* probabilities we calculated before are not useful here, and we need to calculate new ones, namely $P(GlassBr, Fire, Screams|Acc)$ and $P(GlassBr, Fire, Screams)$. What if we want to learn of the probability of there being an accident and explosion given that we have these three events? We can see the infeasibility of this approach as for every variable in our system and for every combination, we must calculate a large amount of probabilities. Again, we could use lower and upper probabilities, but this would leave us in a similar state of ignorance as we saw in Chapter 2.

In the Riot Detection Task, where we want to infer the presence of Riots given the presence of the other concepts, we would have to calculate the a priori probabilities of there being a riot given that there are different combinations of the other concepts. Additionally, these probabilities would only be useful for detecting riots, and if we want to train the model for other of the events, we would have to learn some more probabilities.

We propose to combine with the TBM the *a priori* probabilities a concept entails with any other learned *a priori* probabilities of other concepts to gain a better understanding of the scene. This would allow us to not only combine the known *a priori* probabilities, but also the probabilities can be in different domains (as in, dealing with different variables – For example one sensor dealing with crowds and gunshots, and other sensor dealing with crows and vandalism) and we would still be able to combine

them. We would also be able to query the system for the presence of any combination of events. All of this calculating only the *a priori* probabilities of the different events. Our approach consists of modelling the *a priori* probabilities as Belief Potentials which we can then combine using the Dempster's Rule of Combination. In the next sections we will explain our approach.

3.2 Proposed Approach

As we saw previously, belief in the TBM is modelled with Potentials. Each Potential has a collection of Focal Elements which support specific Configurations with a given Belief Mass. We propose to use the Potentials to model the knowledge of each *a priori* probability, which can then be later combined to create a single potential with all the combined *a priori* probabilities.

The challenge is then to model the *a priori* knowledge in a way which can be reasoned and combined using the TBM, i.e. as Belief Mass Potentials. Our approach can be split in two parts: in the first part, the *a priori* knowledge is computed (only for pairs of concepts), modelled as Focal Elements and the Potentials constructed and combined; in the second part, we get the incomplete/imprecise knowledge from the world and try to infer its state given the precomputed Potentials. The explanation of the two stages will be carried out using the riot detection task, which is another of the contributions of this work. The dataset consists of real CCTV footage from the 2011 London riots. The footage was annotated for the presence of six different concepts related to the riot detection, namely fire, crowd, vandalism, running, people with face covered, and riot. More information on the dataset and the experimental set up can be found on Chapter 5.

	running	face covered	crowd	fire	riot	vandalism
running	1	0.536739	0.451844	0.152078	0.451844	0.516101
face covered	0.04505	1	0.721639	0.243126	0.708875	0.839714
crowd	0.046188	0.878868	1	0.304851	0.949967	0.93915
fire	0.043178	0.822424	0.846736	1	0.845032	0.999127
riot	0.04862	0.908792	1	0.320261	1	0.986849
vandalism	0.046137	0.894354	0.821315	0.314583	0.81985	1

Fig. 3.1: Probabilities matrix for the Riot Detection problem

3.2.1 Training

First we must train the system to create a model of the world which will be applied to a given instance. The training involves learning from a particular dataset, the probabilities of the events and the computation of the global potentials for each event.

We will first explain the process and then illustrate with an example. In this phase, the *a priori* probabilities are first calculated from a training set. On the Riot Detection task, and with the training data used, we get the probabilities matrix presented in Figure 3.1. In this matrix, the columns represent the entailed concepts and the rows the concept that entails.

From this, we can see that some concepts are related. For example, the presence of riot entails always the presence of a crowd. And running is not entailed strongly by other concepts. Riot in particular is highly entailed by most other concepts given that this particular dataset is specifically designed for this task. Figure 3.2 presents a particularly eventful frame with riot. This frame has crowds, people with the face covered, vandalism, and of course, riot. There are parts in this sequence where people also run.

After the probabilities have been computed, we construct potentials for each pair of concepts and combine them successively for each concept to reach a global combined potential. Each Potential's Domain will include only the variables involved, as the TBM allows us to combine Potentials from different Domains. The potential must be con-

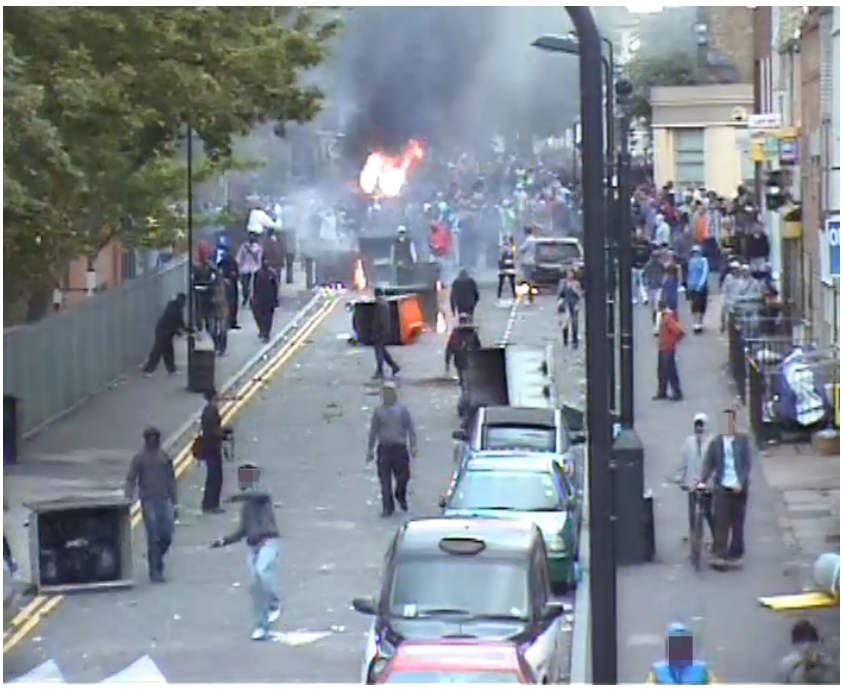

Fig. 3.2: Example of a frame with riot

structed in a special way to encode correctly the knowledge.

3.2.1.1 Designing of Global Potential

One of the most important aspects of this approach is the design of the global potential for each event. A wrong global potential means that the knowledge will not be correctly encoded and thus lead to misleading or wrong results.

A correct global potential will encode correctly the knowledge of each pair of events. Let us start with the pair of concepts "face covered" and "fire". According to Table 3.1, "face covered" entails the concept of "fire" with a probability of 0.243126. This means that if in a certain frame we learn of the presence of "face covered", there is a 0.243126 probability of there also being "fire". To put in TBM's terms, and if we follow Bayes' subjectivist view, this means that our belief that there is "face covered" is 1 (as it is the concept we are certain of) and our belief that there is "fire" is 0.243126.

From this we can see that:

- All of the focal elements must fully and exclusively support the entailing concept, "face covered". This means that its negation will not be present in any of the focal elements of the potential.
- The support or belief for the entailed concept must be equal to the probability. In our case, it means that one of the focal elements must support exclusively (on this variable) "fire" with a belief of 0.243126.
- The beliefs of all the focal elements still must add up to 1, so the remaining support (0.756874) must be given to the remaining focal element(s).

Having this in mind, we have a partial potential for this concept:

{{`face covered, fire`}}[0.243126] {{`face covered, ?`}, `?`}[0.756874]

This global potential already satisfies the requirements, as the belief of "face covered" is 1 and the belief of "fire" is 0.243126. Now the issue remains of what to give the rest of the belief support in the second or more focal elements. We know that this support cannot be given to the negation of "face covered", and the support that was going to be given only to "fire" has already been taken care of. We only have the option left of the negation of "fire".

If we go back to the interpretation of the data, the remaining is not supporting directly the negation of "fire", it is simply information that we do not have available to distinguish between the concept and the negation. This means that the plausibility (or upper probability) of "fire" is still 1. This leaves us with the conclusion that the remaining support must me given to "fire" and its negation. Leaving us with the following global potential:

{{`face covered, fire`}}[0.243126]

{{`face covered, fire`}, {`face covered, no fire`}}[0.756874]

We can also reach the same global potential if we model the problem as the combination of two single-variable potentials on different domains. As we said before, our belief that there is "face covered" is 1 (as it is the concept we are certain of) and our belief that there is "fire" is 0.243126. We could then construct this knowledge as two separate potentials on the two different variables:

Potential for Face Covered:

{{{face covered}}[1]}

Potential for Fire:

{{{fire}}[0.243126], {{fire}{no fire}}[0.756874]}

Now if we apply the fusion operation on this two Potentials (the combination of the extension of the two potentials on the combined domains), we get:

Extended Potential for Face Covered:

{{face covered, fire}{face covered, no fire}}[1]}

Extended Potential for Fire:

{{fire, face covered}{fire, no face covered}}[0.243126]

{Ω}[0.756874]

If we combine this two potentials which are now in the same domain, we get the exact same global potential as before:

{{face covered, fire}}[0.243126]

{{face covered, fire}, {face covered, no fire}}[0.756874]

To summarise, each initial Potential has two Focal Elements: one with a single Configuration with the two concepts (the entailing and the entailed) and a mass equal to the *a priori* probability of the two concepts, and the other with two Configurations: one is

the two Concepts and the other is the entailing concept and the negation of the entailed concept with the complement of the probability as the mass. With this we get lower and upper probabilities that are consistent with the knowledge, as in the beliefs of the two concepts appearing together is the *a priori* probability, but the plausibility is 1.

Effectively we can see that if we learned that there is people with the face covered in the scene, the belief of the system that there is also a fire is 0.243126 (as in the traditional probabilistic world) and the plausibility is 1. The belief that there is no fire is 0, but the plausibility is 0.756874. The belief and plausibility of "face covered" is 1, meaning that we are certain that there is "face covered" as it is what we have just learned.

We create now the second initial Potential for the pair of concepts "face covered" and "riot":

{{`face covered, riot`}}[0.708875]

{{`face covered, riot`}, {`face covered, no riot`}}[0.291125]

3.2.1.2 Combining the Potentials for Each Concept Pair

We combine the two initial Potentials and get the following global Potential for the concept "face covered":

{{`face covered, fire, riot`}}[0.17234594325]

{{`face covered, fire, riot`},{`face covered, fire, no riot`}}[0.07078005675]

{{`face covered, fire, riot`},{`face covered, no fire, riot`}}[0.53652905675]

{{`face covered, fire, riot`},{`face covered, no fire, riot`},{`face covered, fire, no riot`},{`face covered, no fire, no riot`}}[0.22034494325]

The Potential is still valid, as we can see that all the masses add up to 1. More importantly, it still encodes correctly our knowledge of the world, as if in a particular

scene we learn only that there is people with the face covered, we can automatically infer the entailment of the other two concepts we have trained so far. If we query this Potential, we can see that the original beliefs and plausibilities have not changed. But we can now query for example the belief that there is fire and riot, which in this case is 0.17234594325, as only the first Focal Element supports fully the two Concepts; the plausibility of the presence of the two concepts is still 1. Again, the belief that there is people with the face covered is 1 as it is the concept we know for sure is present in the scene.

We repeat this process for each pair of concepts for the concept we are training (i.e. "face covered" and "running", "face covered" and "crowd", etc) combining the initial potentials with the concept's global Potential. In the end, this global Potential will carry all the knowledge the concept "face covered" entails. We repeat this process for all the concepts and end up with a collection of global Potentials, one for each concept.

There are two situations that must be addressed. The first one, when the probability is equal to zero, and the way to handle it depends on each domain and if we are using Open World or Closed World assumption. Under the Closed World assumption, this means that the concept does not support the other concept, in which case the Potential for this pair of concepts is just the entailing concept with the negation of the entailed concept and a mass of 1. Under the Open World assumption, it means that we just don't know about the entailment of this two concepts, in which case we simply don't add this combination of concepts to the global potential (i.e. the initial Potential for that concept is not combined with the global Potential). The second one is in the case when the probability is 1, meaning that the presence of a concept fully entails the presence of another concept, in that case the second Focal Element is not added and the first one gets the full belief (1).

3.2.2 Feeding the Knowledge Base

Once the global Potentials for the concepts have been computed, knowledge can be acquired from the world we are working with. For the Riot Detection task, and for the sake of brevity, imagine we have trained the model with only three concepts: "crowd", "running", and "riot". In this example, the global Potentials for "crowd" and "running" are:

crowd:

{{crowd, running, riot}}[0.043877075796]

{{crowd, running, riot},{crowd, running, no riot}}[0.002310924204]

{{crowd, running, riot},{crowd, no running, riot}}[0.906089924204]

{{crowd, running, riot},{crowd, no running, riot},
{crowd, running, no riot},{crowd, no running, no riot}}[0.047722075796]

running:

{{running, crowd, riot}}[0.204163000336]

{{running, crowd, riot},{running, crowd, no riot}}[0.247680999664]

{{running, crowd, riot},{running, no crowd, riot}}[0.247680999664]

{{running, crowd, riot},{running, no crowd, riot},
{running, crowd, no riot},{running, no crowd, no riot}}[0.300475000336]

Suppose that we learn a particular scene has "crowd" and "running" in it. We first create a Potential for that frame with a copy of the "crowd" concept global Potential. This will include in the knowledge base not just the concept from the detector, but the

associated probabilities that come with it. If another detector indicates that the same frame has "running" in it, we combine the Potential for this concept with the frame's previous global Potential, creating a new global Potential for the frame with all the knowledge that we have about that frame. This includes the belief that there is riot in the scene given these two concepts, even though we have not learned about riot from any sensor in this frame. The resulting Potential for that frame would be:

{{`running, crowd, riot`}}[0.972574110852]

{{`running, crowd, riot`},{`running, crowd, no riot`}}[0.027425889148]

Please note that since we already learned that running and crowd are present, all the Focal Elements containing the negation of those concepts are no longer present in the model. In this model, the Belief that there is riot is 0.972574110852 (because one of the concepts strongly supports this hypothesis) and the Plausibility is 1.

3.2.3 Unreliable Sensors

This can be easily adapted to the case when we have unreliable sensors by normalising the mass of the two Focal Element by the confidence score given by the detector and adding a third Focal Element for the entire frame of discernment (Ω) with a mass equal to the complement of of the confidence score. In other words, the configurations of this third Focal Element are a Cartesian product of the two concepts and their negations and assigning to it the remaining mass of the Potential. This allows us to represent the case when the sensor is wrong and we can not tell the actual state of the world. i.e. the Potential for the previous example of "face covered" and "fire" would be:

{{`face covered, fire`}}[0.243126]

{{`face covered, fire`}, {`face covered, no fire`}}[0.756874]

{Ω}[0]

And then when we are populating the knowledge base and a detector tells us that a frame has a person with their face covered with a certainty of 80%, the global Potential for the frame becomes:

{{`face covered, fire`}}[0.1945008]

{{`face covered, fire`}, {`face covered, no fire`}}[0.6054992]

{Ω}[0.2]

Indeed we can see that if we query this Potential it represents precisely our knowledge about the model, as the belief that there is a person with the face covered is 0.8, but the plausibility is 1. The belief that there is a fire is now 0.1945008 and the plausibility is still 1.

3.3 Concluding Summary

In this Chapter, we have presented our approach to combine *a priori* probabilities using Dempster's Rule of Combination. This requires first that the priors be modelled as Belief Potentials. In the first part of the Chapter, we have shown the limitations of traditional Bayes' Rule and why our approach is useful.

This approach has some other applications which we do not intend of exploring on this work, but are left for future works. For example since a knowledge based modelled under this approach includes knowledge of all the concepts for which the model has been trained, it works not just for concepts for which there is no detector yet, but also for those for which there might be a detector too. This is desirable in the case when one of the detectors fails, we can still infer a degree of belief of the missing concept given that we know the presence of other concepts that entail such concept. For example if we had a gunshot detector and a firearm detector, and in a given frame we learn that a gunshot was heard, and there was a firearm too but the firearm detector was offline at

that moment, we still have a certain belief that there was a firearm in the scene.

The application that we do explore on this work is that high level events for which there is no detectors yet, can be inferred from the presence of other concepts that entail such high level concept. In particular, we use this approach together with the Semantic Reasoner presented in Chapter 4 to detect riots given that we know the presence of other concepts in the scene. Chapter 5 presents the results of this task.

Chapter 4

TBM for the Semantic Web

From the previous chapter we can see that the proposed approach for combining *a priori* probabilities with the TBM can lead to Potentials with large graphs. For example, a single global Potential for a Domain trained with 6 Variables, can be composed of a few dozens Focal Elements, each Focal Element with multiple Configurations and each with multiple Elements. It is crucial then to develop a TBM reasoner that is scalable and efficient. Coupled with the fact that we will be using it with Multimedia data (which tends to be large), it becomes apparent that the system must be as efficient as possible.

The next contributions of this work are directly related to the following requirements presented in Chapter 1:

- **R1** as we are using the TBM to deal with imprecise and uncertain information. This also answers successfully **RQ3**.
- **R2** because we are including parallel heuristics to make the reasoning process scalable with large datasets. Here we also answer **RQ4**, as these heuristics are sufficient to work with thousands of triples in the knowledge base.
- **R3** as we are using Jena's open source framework to create an RDF and OWL compliant ontology and reasoner which can connect to any standar RDF or OWL

repository.

We first present the Ontology for the TBM. To introduce its concepts, we will be using the two player game of Cluedo introduced in [1] where we are tasked to find the murderer, the weapon, and the room where the murder happened. We define our world in a simple ontology with three classes: Murderer (with instances "Colonel Mustard", "Miss Scarlett", and "Mrs. Peacock"), Room (with instances "Dining Room" and "Kitchen"), and Weapon (with instances "Dagger" and "Candle Stick"). A simplified version of the triples in the ontology in N3/turtle format is as follows:

```
@prefix :    <http://example.org/CLUE.owl#> .
@prefix owl:  <http://www.w3.org/2002/07/owl#> .
@prefix rdf:  <http://www.w3.org/1999/02/22-rdf-syntax-ns#> .

:Murderer rdf:type owl:Class .
:Room     rdf:type owl:Class .
:Weapon   rdf:type owl:Class .

:ColMustard     rdf:type :Murderer .
:MissScarlett   rdf:type :Murderer .
:MrsPeacock     rdf:type :Murderer .
:DiningRoom     rdf:type :Room .
:Kitchen        rdf:type :Room .
:CandleStick    rdf:type :Weapon .
:Dagger         rdf:type :Weapon .
```

The first player has a strong subjective belief of 80% (Belief Mass) that the murderer is "Colonel Mustard" and the murder happened in the "Kitchen". He is not sure about the weapon. The second player believes also with a subjective 80% that the murderer is "Colonel Mustard" but the weapon is "Candlestick" and he is not sure about the room.

4.1 TBM Ontology for the Semantic Web

The ontology is a common vocabulary to represent reasoning problems in a standard way. For this, we identify the important and relevant concepts for the reasoning task and encode them in an OWL ontology.

The name space of both the concepts and the operations of the reasoner is `http://mmv.eecs.qmul.ac.uk/TBM.owl#`. In the remaining parts of this document, we will use the prefix `TBM` for the URIs in this namespace. The ontology is available online at: `http://www.eecs.qmul.ac.uk/~chps3/TBM.owl`

4.1.1 Classes

Following is an introduction to the classes of the TBM ontology. We explain briefly each concept and present an example of assertions in the knowledge base using the example in the domain.

- `TBM:VarDomain` is a set of `OWL:Class` individuals over which the reasoning task is defined. In our game of Clue example, the domain is {`:Murderer, :Room, :Weapon`}, but the domain of the first player is {`:Murderer, :Room`} and the domain of the second player is {`:Murderer, :Weapon`}.

- `TBM:Configuration` is a collection of instances from the domain (one for each variable). It represents a piece of information in the knowledge base. {`:ColMustard, :Kitchen`} is an example of a configuration.

- `TBM:FocalElement` are subsets of the Frame of Discernment (Ω) with non-null belief masses. In other words a collection of configurations on which the potential has some degree of belief that one of the hypothesis is the solution. In our example, one of the focal elements of player one has a single configuration and is {{`:ColMustard, :Kitchen`}}. The other focal element would be the entire

frame of discernment (a Cartesian product of all the instances of the variables in the domain - one configuration for each possible combination) with the remaining belief mass (in this case 0.2 — belief masses are expressed as floating point numbers and must add up to one). As explained before the practice of adding a belief mass to the entire frame of discernment is done to express ignorance, it means the information available does not allow you to differentiate between all the possible hypothesis.

- `TBM:Potential` is a known fact about the task in the frame of discernment. It is a collection of Focal Elements that represent said knowledge. It can be the result of an observation, a measurement, or any other way to acquire knowledge. In our example, we have two sources of information corresponding to each player. The first player's potential consists of his two focal elements and their respective belief masses. i.e. {{`:Murderer, :Weapon`}} with a Belief Mass of 0.8, and {Ω} with a Belief Mass of 0.2.

4.1.2 Object Properties

We now present the object properties of the ontology. They are the relationships between the different concepts in the TBM ontology and with the concepts of the problem's domain.

- `TBM:hasVarDomain` the domain of this property are `TBM:Potential` and `TBM:FocalElement`. The Range is `TBM:VarDomain`. It allows to specify the domain of a Potential or a Focal Element.

- `TBM:hasVariable` has domain `TBM:VarDomain` and range any RDF resource of type `OWL:Class`. It encodes which variables belong to a given domain.

- `TBM:hasFocalElement` with domain `TBM:Potential` and range `TBM:FocalElement`. Allows to link a potential with all its Focal Elements.

- `TBM:hasConfiguration` has a domain of `TBM:FocalElement` and range `TBM:Configuration`. It links the Focal Elements with its Configurations.

- `TBM:hasElement` domain is `TBM:Configuration` and range is any `OWL:NamedIndividual` of a type specified in the `TBM:VarDomain`. It encodes the actual individuals that belong to a configuration.

4.1.3 Data Properties

There is only one Data Property in the ontology.

- `TBM:hasMass` has a domain of `TBM:FocalElement` and a range of `xsd:double`. It represents the belief that a given Focal Element holds the answer. All of the masses of the Focal Elements of a Potential must add up to 1.

The knowledge of the domain must then be encoded in OWL triples and using the TBM ontology. A simplification of the graph for the potential of player one can be seen in Figure1.1.

Some of the simplifications for space and clarity were: the types of some of the nodes, the domain node of the Potential and the Focal Elements, and the representation of the frame of discernment (Ω), which in reality should be several configurations, one for each possible combination of the instances of the variables in the domain. The focal elements and configurations are modelled as B-nodes as it is not necessary to name them. Player two's potential has a similar graph but the domain is different.

Under this paradigm, a problem domain would define its own concepts and use potentials from the TBM ontology to represent a specific piece of knowledge. For example for a multimedia surveillance ontology, the variable would be the classes of events and objects. The visual analysis would feed the knowledge base with the levels of confidence of the detection task (i.e. a given blob is a person with a confidence of 70%).

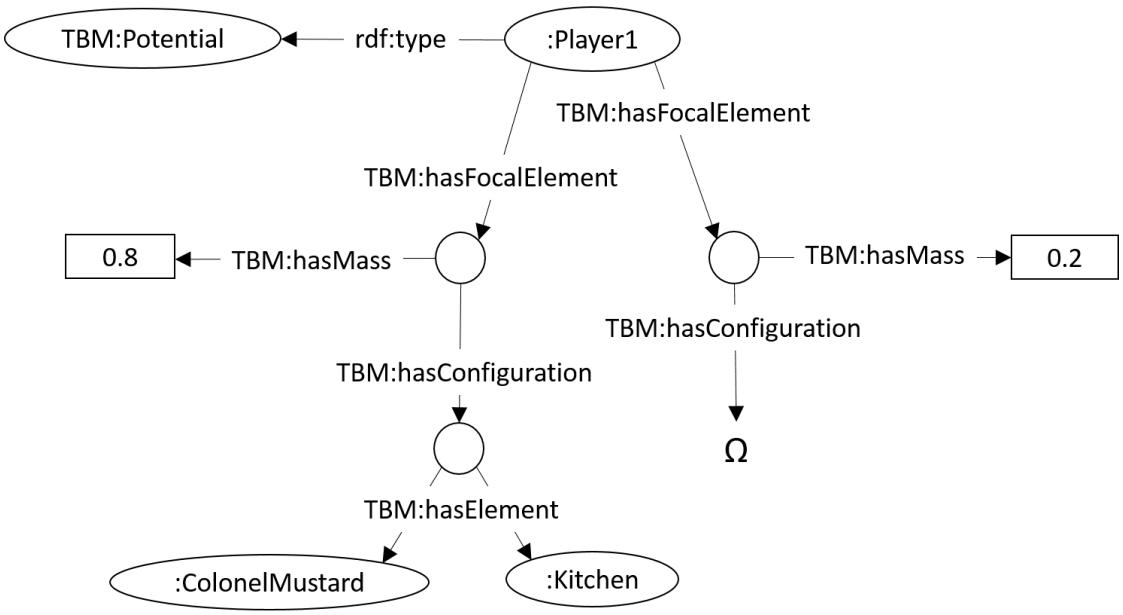

Fig. 4.1: TBM graph of the knowledge from player one

4.2 TBM Reasoner for the Semantic Web

Our second contribution consists of the actual engine to perform the reasoning using the TBM. It consists of implementations for the semantic web of the different operations and functions defined in the TBM applied to ontologies and using the ontology defined previously. The mathematical definitions of these operations were presented in Section 2.3.

As seen in [57], implementations of the algorithms depend largely on the data structures used to represent the potentials. The most computationally efficient (but not memory efficient) implementation being the bitset [24].

The main idea behind representing the focal elements as bitsets is that all the possible configurations have a specific bit in a globally ordered bitset. If for example we had a domain with two variables $\{A, B\}$, and A had instances $\{A', A''\}$, and B had instances

$\{B', B'', B'''\}$. Then the ordering of the bitset could be:

$$\{\{A', B'\}, \{A', B''\}, \{A', B'''\}, \{A'', B'\}, \{A'', B''\}, \{A'', B'''\}\}$$

So if you wanted to represent for example a focal element with the configurations $\{\{A', B''\}, \{A', B'''\}, \{A'', B'''\}\}$, the bitset would be 011001.

The main advantage of this approach is that the operations can be very efficiently implemented using logical bitwise operations. For example the combination of two focal elements can be performed with a bitwise "and". However, The focal elements are always the same size regardless of the number of configurations present in it, and this size grows exponentially with each new instance.

Other previous proposals include List Representation, Disjunctive or Conjunctive Normal Form. We propose a novel way to use graphs to represent the focal elements and potentials. As such, we had to devise new algorithms for the TBM operations and the graph representation of potentials. We now proceed to present our contribution.

The current implementation of the reasoner is achieved in Java using the Jena Reasoner API [58] for ontologies. The source code is hosted on Github at `https://github.com/CesarPantoja/TBM`. We will continue with the example from the previous section to introduce the different operations in the reasoner.

4.2.1 Extension

The Extension operation takes a reference to a Focal Element which is in a given domain, and a target domain. It returns a new Focal Element which takes all the configurations in the given Focal Element and adds instances of the variables not present in the original Domain. The pseudo-code of this method is presented in Algorithm 1. It works by first cloning the input Focal Element and setting its Domain to the input Domain on lines 1 to 3. If there are no differences between the input Domain and the Focal Element

Domain's variables, the clone is returned. Otherwise for each different variable, on line 6, the result Focal Element's Configurations are replaced with the Cartesian product of the Configurations with the instances of the new Variable.

Algorithm 1 extension(fe, d)

Require: {fe `rdf:type TBM:FocalElement`}
Require: {d `rdf:type TBM:Domain`}
Require: fe.domain $\subseteq$ d
1: diff $\leftarrow$ d $-$ fe.domain
2: result $\leftarrow$ clone fe
3: result.domain $\leftarrow$ d
4: **if** diff $<> \oslash$ **then**
5: **for** var $\in$ diff **do**
6: result.configs $\leftarrow$ result.configs $\times\{x|x$ `rdf:type` var$\}$
7: **end for**
8: **end if**
9: **return** result

When the two Domains are not different, this method is of time complexity $O(m \times n)$ where n is the number variables in the Domain and m is the number of Configurations. This is because the clone operation still has to go through all the elements and create new relations with the new Configurations. If the two Domains are different, the time complexity is $O(n \times m)$ where n is the number of different instances in the different variables and o is the number of Elements in all Configurations of the Focal Element.

4.2.2 Combination

The next operation is Dempster's Rule of Combination, which is the most complex operation of the reasoner. It's input are two potentials and it returns a combined potential which represents the combined belief. The combination can happen between potentials in different domains, as we also apply the fusion operation (extend the two potentials with the combined domain). In our example, we want to combine the knowledge of the first player which is in the domain {`:Murderer, :Room`} with the knowledge of the second player in the domain {`:Murderer, :Weapon`}. The combination will produce a new Potential which includes the knowledge from both potentials. The idea is to take

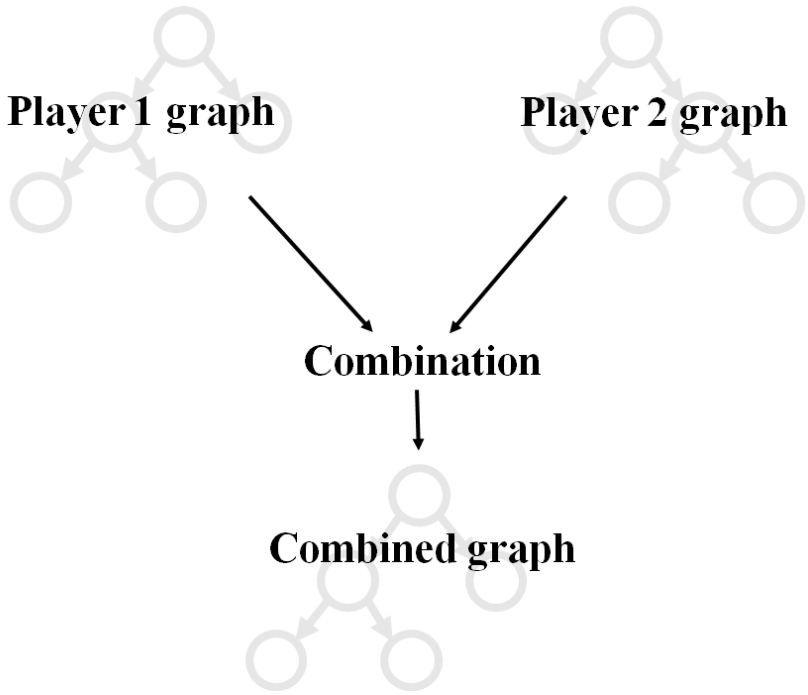

Fig. 4.2: Combination of the knowledge of the two players

the two Potentials' graphs and create a new one as seen on Figure 4.2.

We must clarify again here that we are working under the close world assumption, which is why on the case of a conflict, the other masses are normalised with the conflict so the potential's masses add up to 1 again. Algorithm 2 presents the pseudo-code for this operation.

In this implementation, on lines 1 to 11, we first extend all the Focal Elements in the two Potentials to have the same variables (fusion operation), effectively placing them in the same Domain. Then, both collections of extended Focal Elements are explored with two nested loops (lines 13 and 14). In the inner loop, a new Focal Element is created in line 15 and the Configurations of the Focal Elements are compared (lines 17 to 23). If the two Configurations have the same elements, they are added to the result Focal Element (Which is done in line 20). Since we are working under the closed world assumption, if none of the Configurations are equal, the combined Mass is added to the conflict (line 25), otherwise it is set on the result Focal Element and added to the result Potential

Algorithm 2 combine(p1, p2)

Require: {p1, p2 `rdf:type TBM:Potential`}

```
 1: result ← new Potential
 2: comb_domain ← p1.domain ∪ p2.domain
 3: result.domain ← comb_domain
 4: ext_fe1 ← {}
 5: for fe ∈ p1.f_elements do
 6:    ext_fe1 ← ext_fe1 ∪ extend(fe, comb_domain)
 7: end for
 8: ext_fe2 ← {}
 9: for fe ∈ p2.f_elements do
10:    ext_fe2 ← ext_fe2 ∪ extend(fe, comb_domain)
11: end for
12: conflict ← 0
13: for fe1 ∈ ext_fe1 do
14:    for fe2 ∈ ext_fe2 do
15:       res_fe ← new TBM:FocalElement
16:       res_fe.domain = comb_domain
17:       for c1 ∈ fe1.configs do
18:          for c2 ∈ fe2.configs do
19:             if c1 = c2 then
20:                res_fe.configs ← res_fe.configs ∪ c1
21:             end if
22:          end for
23:       end for
24:       if res_fe.configs = ∅ then
25:          conflict ← conflict+(fe1.mass ∗ fe2.mass)
26:       else
27:          res_fe.mass ← fe1.mass ∗ fe2.mass
28:          result.f_elements ← result.f_elements ∪ res_fe
29:       end if
30:    end for
31: end for
32: if conflict> 0 then
33:    for fe ∈ result.f_elements do
34:       fe.mass ← fe.mass / (1−conflict)
35:    end for
36: end if
37: return result
```

(lines 27 and 28). At the end, if there is a conflict, all of the Masses are normalised to add up to 1 again on lines 32 to 36.

Time complexity for this operation is $O(n \times m)$, where n and m are the total num-

ber of configurations in the extended Focal Elements of both potentials. In our case, the combination of the two Potentials results in a Potential with four Focal Elements: {{`:ColMustard, :Kitchen, :CandleStick`}} with a Belief Mass of 0.64, {{`:ColMustard, :Kitchen, :Dagger`}{`:ColMustard, :Kitchen, :CandleStick`}} with a Belief Mass of 0.16, {{`:ColMustard, :DiningRoom, :CandleStick`}{`:ColMustard, :Kitchen, :CandleStick`}} with a Belief Mass of 0.16, and {Ω} with a Belief Mass of 0.04.

4.2.3 Belief

We can at any point in time interrogate a Potential to assess our knowledge. The Belief Functions allow us to do just that. The first such function is *Belief*, which we can know intuitively to be the amount of support that a subset of Ω holds the truth. The pseudo-code for this function is presented in Algorithm 3

Algorithm 3 belief(p, query_fe)

Require: {p `rdf:type TBM:Potential`}
Require: {query_fe `rdf:type TBM:FocalElement`}
Require: query_fe.domain ⊆ p.domain
1: belief ← 0
2: **for** fe ∈ p.f_elements **do**
3: **if** fe.configs = query_fe.configs **then**
4: belief ← belief + fe.mass
5: **end if**
6: **end for**
7: **return** belief

This algorithm works simply by comparing the input Focal Element with the Potential's Focal Elements. The input Focal Element's Domain must be a subset or equal to the Potential's Domain. If any of the Focal Elements have the same elements (not including instances of variables not in the input's Domain), the mass is added to the overall Belief which is returned at the end. The time complexity of this algorithm is $O(n)$ where n is the number of total configurations in the Potential. In our example, we might be interested in knowing the belief that {`:ColMustard, :Kitchen, :CandleStick`} is the solution to our problem. In this case, there is only one Focal Element which supports

this hypothesis and has a Belief of 0.64, value which is returned by this operation.

4.2.4 Plausibility

This is the amount of support to a given hypothesis, including the Belief of Focal Elements which support it at least partially. Algorithm 4 presents the pseudo-code for this operation.

Algorithm 4 plausibility(p, query_fe)

Require: {p `rdf:type TBM:Potential`}
Require: {query_fe `rdf:type TBM:FocalElement`}
Ensure: query_fe.domain $\subseteq$ p.domain

```
1: pls ← 0
2: for fe ∈ p.f_elements do
3:    for conf ∈ fe.configs do
4:       for conf_query ∈ query_fe.configs do
5:          if conf_query ⊆ conf then
6:             pls ← pls + fe.mass
7:          end if
8:       end for
9:    end for
10: end for
11: return pls
```

In this algorithm, all the Configurations are compared, and if any of the query Configuration is a subset of one of the Potential's Configuration, the mass of its Focal Element is added to the variable pls, which is returned at the end. The time complexity of this algorithm is also $O(n)$ where n is the total number of Configurations. Going back to our example, the plausibility that the hypothesis {`:ColMustard, :Kitchen, :CandleStick`} is the solution, is equal to 1. This is because all of the Focal Elements support it at least partially.

4.2.5 Doubt and Ignorance

The remaining two functions are defined in terms of the previous two. *Doubt* is the belief which will never be assigned to the hypothesis. It is defined as the complement

of the plausibility, or subtracting the plausibility from 1. *Ignorance* is belief which is not assigned exclusively to a hypothesis, in which case we can not distinguish which from the other hypothesis is the correct one. It is the difference between the plausibility and the belief. In our example, the Doubt and Ignorance of {`:ColMustard, :Kitchen, :CandleStick`} are 0 and 0.36 respectively.

Another of our contributions is the implementations of the TBM belief functions (belief, plausibility, ignorance, and doubt) into Jena's SPARQL engine. This means that there are two ways to access the belief operations: programmatically through the Java API or as functions in SPARQL. In the Riot Detection task, imagine we have a knowledge base for CCTV knowledge, and we want to retrieve all the shots where the belief that there is a riot is greater than 80%. An example of a very simple SPARQL query to achieve this could be:

```
PREFIX :    <http://example.org/CCTV#>
PREFIX TBM: <http://mmv.eecs.qmul.ac.uk/TBM.owl#>
SELECT ?shot
WHERE {
    ?shot a :Shot .
    FILTER(TBM:bel(?shot, :Riot) > 0.8)
}
```

Empirically we can see that the most complex operation is the Combination, and that one of the main factors affecting the complexity is the number of Focal Elements, because for each one of them, there is minimum one Configuration with one Element, and calculations have to be performed for each one of them. Our performance evaluation objectively confirms this.

4.2.6 Design and Architecture

The implementation of the reasoner follows closely that of the rest of Jena. In Jena, a graph is called a model. There are different kinds of models, but in Jena they are all represented by the `Model` interface. Jena, and subsequently our reasoner, follows mainly two design patterns:

- Models are created using the **Factory** pattern [59] through the classes `ModelFactory` in Jena and `TBMModelFactory` in the TBM reasoner. In the Factory pattern, objects of different classes are created using some *Factory* methods without having to know beforehand the type of the object you want to create. The responsibility of the Factory classes mentioned before is to create different kinds of models. For example Jena's `ModelFactory`, creates memory or TDB backed models, with or without inference, RDFs or OWL, and more. `TBMModelFactory` creates a `TBMModel`, but this could be extended to create TBM reasoners with different capabilities in the future.

- Most of the other data model classes follow the **Bridge** design pattern [59]. Under this pattern, the Class itself is the abstraction and what it does is the implementation. The abstraction and the implementation are then decoupled, generally by interfaces or abstract classes and implementations of those interfaces. This allows greater flexibility when there are multiple implementations of the same abstraction and/or when the implementation varies frequently. For example, in Jena, all models are represented by the `Model` interface, and different kinds of models implement this interface. This means that the abstraction of "model" is decoupled from the actual implementation, allowing us in our case to create a type of model for the TBM. Although in the TBM reasoner there is only one implementation for the classes, we remained within this design pattern to maintain consistency with Jena's architecture and to allow easy extensibility in the future. In the TBM reasoner, `TBMModel` is an interface (which extends from Jena's `Model` interface), and

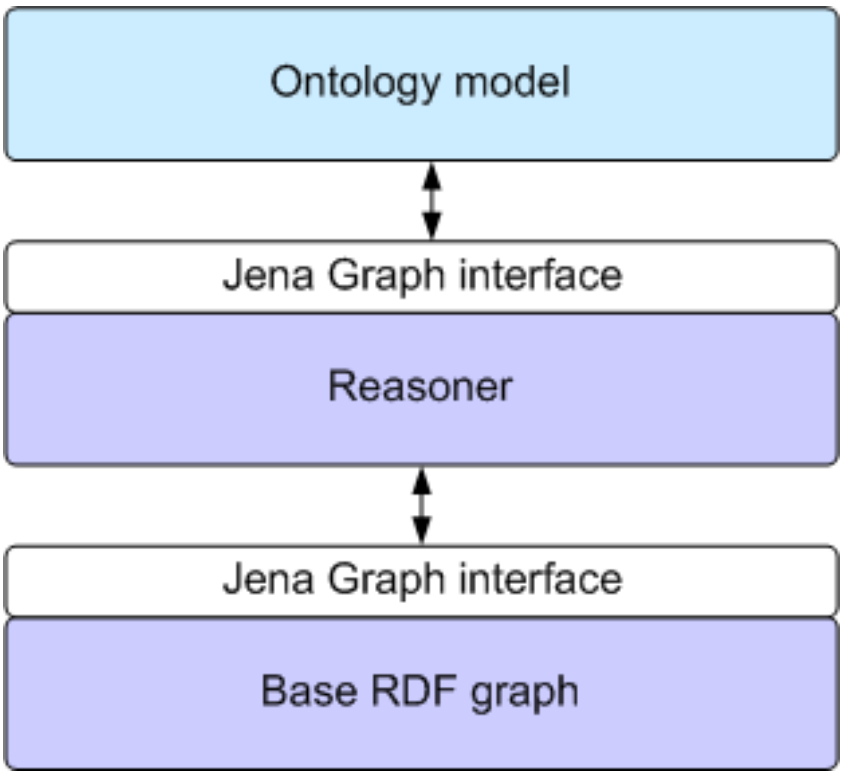

Fig. 4.3: Jena's Reasoner Model Architecture [3]

the actual implementation of the model is done in the class `TBMModellImpl`. The same applies for the other classes like `TBMPotential`, `TBMVarDomain`, and son on.

Another characteristic of Jena's architecture is the nesting of models. This means that models can be based on models and can serve as bases for other models. For reasoners this means that the model they create appears to contain the inferred tuples along with the tuples asserted in the base model, and it can be queried in the same way as any other model. Figure 4.3 presents how this is achieved in a particular model.

For the TBM reasoner, this means that it can use any other type of model as a base, and the reasoning would be performed on all the tuples (including the inferred ones). And it can also serve as a base for any other kind of model and the reasoning of the other model would be performed including the TBM inferred tuples. This allows us to for example use the TBM reasoner on a model where new class assertions have been inferred through OWL, or use an RDFs reasoner on a model where TBM tuples have

been inferred.

The source code is hosted on Github in a public repository at `https://github.com/CesarPantoja/TBM` and can be freely cloned or downloaded.

4.3 Parallel Heuristics for the Semantic TBM Reasoning Engine

One of the main drawbacks of the TBM is its complexity [24] because of the great number of operations that have to be performed, but it is dependant on the data structure used. The most computationally efficient method for representing the Focal Elements are the bit sets [24]. But great computational performance is achieved in exchange of a great memory footprint. For example, it would take 2^N bits to store a single Focal Element in a domain of N binary (with two instances) variables, no matter the number of Configurations in the given Focal Element.

In our Riot Detection Task, we have 6 binary Variables, one for each concept. This means that a focal element would take $2^6 = 64$ bits (8 bytes). If we take into account that the initial Potential for a frame has two Focal Elements, and that the number of Focal Elements grows exponentially every time we combine the Potential of a new concept, after combining 5 Potentials we have that a single Potential could be as much as $8 \times 2^5 = 256$ MB. And furthermore, if we take into account that a single CCTV shot can have potentially thousands of frames, we quickly realise the impracticality of modelling our problem as binary sets. A second drawback to this approach is the lack of interoperability with any other system.

Modelling the problem in a Semantic Repository allows us to use Jena's indexing capabilities. The key is the URL of the relevant Resource and the repository takes care of the indexing for fast access of the tuples. But this then leads to an explosion in the number of computations that must be performed, admittedly the computations

themselves are simple and they can be performed independently, which is why we propose a parallel implementation of the Combination operation of the TBM. We use a fork-join model with a work-stealing pool to ensure maximum utilisation of the CPU. In basic terms, a work stealing pool consists of a thread queue for each processor in the system. Each thread can itself spawn new threads which are queued in the processor's queue. When one of the processors finishes with its queue, it can "steal" work from the other processors' queues [60].

The goal is to parallelise as many operations as possible, and this is achieved in two stages. The first stage is the combination of the Focal Elements. In this stage, all the Focal Elements must be compared, for each possible combination of pair Focal Elements, a new thread is created. Suppose we have potential P' with focal elements $\{F1', F2', F3', ..., Fn'\}$ and potential P'' with Focal Elements $\{F1'', F2'', F3'', ..., Fn''\}$, then the threads would fork as combining the following focal elements:

$$\{F1', F1''\}, \{F1', F2''\}, \{F1', F3''\}, ..., \{Fn', F(n-1)''\}, \{Fn', Fn''\}$$

The second part is the combination of the Configurations inside each Focal Element combination thread. For example, if we had Focal Element F' with Configurations $\{C1', C2', C3', ..., Cn'\}$, Focal Element F'' with Configurations $\{C1'', C2'', C3'', ..., Cn''\}$, then there would be one thread for each pair like so:

$$\{C1', C1''\}, \{C1', C2''\}, \{C1', C3''\}, ..., \{Cn', C(n-1)''\}, \{Cn', Cn''\}$$

To the best of our knowledge, it is the first time that parallel heuristics have been proposed for the combination operation of the TBM. The parallelisation of the combination operation leads to another set of things to consider, such as data synchronisation issues and other aspects inherent to parallel computing. For this reason, in all the stages, appropriate locks are used to ensure the synchronous access to the Conflict variable and

the data-store from all the threads. We also try to minimise the access to disk in the case of a TDB-backed data stores by caching and performing the operations in memory rather than going to disk every time a Resource is needed. This also ensures the transactional integrity of the data base. In the end, the generated tuples are flushed to disk.

To understand the proposed parallelisation method, let us break down the Combination operation in its logical blocks:

4.3.1 Initialisation

The first part is trivial and does not need parallelisation. This part creates the result Potential (which will be returned by the Combination operation) and extends in memory the focal elements of the two Potentials being combined. Also the Conflict variable is initialised. In Algorithm 5, it corresponds to lines 1 to 12.

4.3.2 Combination of Focal Elements

Here we use the term "Combination" in its algebraic meaning, as the Focal Elements of both Potentials have to be combined in order for its Configurations to be compared. This can be seen in the nested loops in lines 13 and 14 of Algorithm 5. This is the first part were the execution is forked. A new thread is added to the execution queue and the two Focal Elements being combined passed as parameters.

4.3.3 Combination of Configurations

This corresponds to the execution of each Focal Element thread and can be seen in Algorithm 6. First a new Focal Element is created. Then, all the Configurations in both Focal Elements are combined (in its algebraic meaning, as in each possible combination of its configurations gets combined) and this thread, on a new work-stealing queue, forks again, one thread per pair of Configurations. After all the threads have been queued, the

Algorithm 5 combine(p1, p2)

Require: {p1, p2 `rdf:type TBM:Potential`}

```
result ← new TBM:Potential
comb_domain ← p1.domain ∪ p2.domain
result.domain ← comb_domain
ext_fe1 ← {}
for fe ∈ p1.f_elements do
    ext_fe1 ← ext_fe1 ∪ extend(fe, comb_domain)
end for
ext_fe2 ← {}
for fe ∈ p2.f_elements do
    ext_fe2 ← ext_fe2 ∪ extend(fe, comb_domain)
end for
conflict ← 0
for fe1 ∈ ext_fe1 do
    for fe2 ∈ ext_fe2 do
        launch thread(combineFElements(fe1, fe2, comb_domain, conflict, result))
    end for
end for
wait for all threads to finish
if conflict> 0 then
    for fe ∈ result.f_elements do
        fe.mass ← fe.mass / (1−conflict)
    end for
end if
return result
```

forking thread joins with the forked thread, meaning that it awaits for all the subthreads launched by this thread to complete. After successfully joining, a check is performed to see if the result Focal Element has no configurations. If this is the case, it means that there is a Conflict and the conflicting mass is accumulated. However if it is not empty, the resulting Focal Element is added to the result Potential and its mass is updated.

4.3.4 Comparison of Elements in the Configurations

This is the last execution level. It corresponds to the execution of the thread of each Configurations pair. This is were the data is accessed and the actual comparisons are performed so no further forking is performed. Pseudo-code for this thread can be seen in Algorithm 7. In this algorithm, the comparison is quite expensive as the elements of the

Algorithm 6 combineFElements(fe1, fe2, comb_domain, conflict, comb_potential)

Require: {fe1, fe2 `rdf:type TBM:FocalElement`}
Require: {comb_domain `rdf:type TBM:Domain`}
1: res_fe ← new `TBM:FocalElement`
2: res_fe.domain = comb_domain
3: **for** c1 ∈ fe1.configs **do**
4: **for** c2 ∈ fe2.configs **do**
5: launch thread(combineConfigs(c1, c2))
6: **end for**
7: **end for**
8: wait for all threads to finish
9: **if** res_fe.configs = ∅ **then**
10: conflict ← conflict+(fe1.mass ∗ fe2.mass)
11: **else**
12: res_fe.mass ← fe1.mass ∗ fe2.mass
13: comb_potential.f_elements ← comb_potential.f_elements ∪ res_fe
14: **end if**

Algorithm 7 combineConfigs(c1, c2, comb_fe)

Require: {c1, c2 `rdf:type TBM:Configuration`}
1: **if** c1 = c2 **then**
2: comb_fe.configs ← comb_fe.configs ∪ c1
3: **end if**

two configurations are compared until one difference is found. If there are no differences, all the configurations are compared. If the configurations are the same, a clone of the configuration is created and added to the result Focal Element and the thread terminates. This cloning operation is also expensive as it involves iterating through all the Elements, creating a copy and appending it to the resulting Focal Element.

4.3.5 Finalisation

The main thread then joins with the Focal Elements threads. If there was a conflict, normalise the masses of the Focal Elements with the conflict so they add up to 1. The Focal Elements are then committed to disk and the result Potential returned.

4.3.6 Concluding Summary

We have presented the design and development of a Semantic Web framework which uses the TBM to model and reason with imprecise and uncertain information. This framework consists first of an Ontology with the main concepts present in the TBM and the TBM operations. Each Domain encodes the knowledge using the TBM Ontology, asserting and combining the different beliefs, and then uses the operations to assess the combined knowledge of the system.

In the next chapter we will test this framework in the riot detection task. Different *a priori* probabilities of different concepts will be combined to reach a common knowledge of the domain.

Using bit-set representation of the potentials is infeasible for the size of problem we are dealing with. In this regard, we have also proposed a new way to represent Potentials and some parallel heuristics for the TBM to make use of multi-core systems popular today. As we will see in the next chapter the implementation is promising as it scales well as you add more resources, but further improvement is needed.

Chapter 5

Framework Evaluation

We validated the proposed framework in two ways. First in its performance, as it is clear that with the kind of problems we are dealing, we can easily achieve Potentials with thousands of Focal Elements and Configurations. The second validation was its usefulness in detecting riots given the known presence of other events that entail such concept. We present the results of the validations next.

5.1 Performance Evaluation

The performance tests we carried out are based on the test bed presented in [1]. The tests are aimed at measuring the execution time and memory consumption of the Combination operation, with different configurations of Frame of Discernment and randomly generated Potentials of different characteristics. Namely:

- Number of Variables
  - 2 to 12 variables
  - 2 realisations per variable

  - 500 focal elements per potential
  - 100 configurations per focal element
- Number of instances per Variable
  - 2 variables
  - 500 to 1000 realisations per variable
  - 100 focal elements per potential
  - 100 configurations per focal element
- Number of Focal Elements per Potential
  - 2 variables
  - 500 realisations per variable
  - 100 to 463 focal elements per potential
  - 100 configurations per focal element
- Number of Configurations per Focal Element
  - 2 variables
  - 500 realisations per variable
  - 200 focal elements per potential
  - 100 to 250 configurations per focal element

Memory was not considered in the tests as it is tightly tied to the Jena framework. In our tests, we used a Memory-backed repository, so it is expected that memory usage will be high in exchange of fast access to the triples. Jena also allows the use of other forms of persistence, each with different benefits and drawbacks. There are extensive evaluations

of Jena's capabilities with different configurations. Exhaustive tests and comparison of Jena's performance can be seen in [61] [62] [63] and [64].

Only the performance of the combination operation was considered, as it is the most complex operation of the reasoner by a large margin. For this reason, we also did not test the SPARQL operations, as it only includes the belief functions and not the combination operation.

To test the scalability of the system, three tests were performed for each variation. Each with different number of CPU cores enabled, on identical machines: server PCs with 2 $\times$ 6 Core Intel Xeon E5645 (Westmere) processors, and Scientific Linux. The three test were with 6, 8, and the last one the full 12 cores enabled.

Next we show the results of the performance evaluation.

In Figure 5.1 we see the results of varying the number of variables in the combination operation. We see it actually has a higher execution time when there are fewer variables. When there are more variables, execution times are reduced until they reach a more or less stable execution time and adding more variables does not change this much. This is due to the fact that, in this particular test the potentials are generated randomly. This causes that when there are less variables, the Elements in the Configurations are more likely to be equal. This in turn triggers the clone operation which is one of the more resource intensive operations. When two Configurations do not overlap, this step is skipped.

Figure 5.2 shows the impact of increasing the number of instances per variable. We can see that although increasing the number of cores does reduce the execution time, increasing the number of instances has no impact on it. This is due to the fact that the number of instances a variable has, does not affect in any way the size of the graph generated for each potential. Let us illustrate this with the two extreme cases in this test: 2 Variables, 100 Focal Elements, 100 Configurations per Focal Element, but one has 500 Instances per Variable and the other has 1000. In the two cases, a graph will be

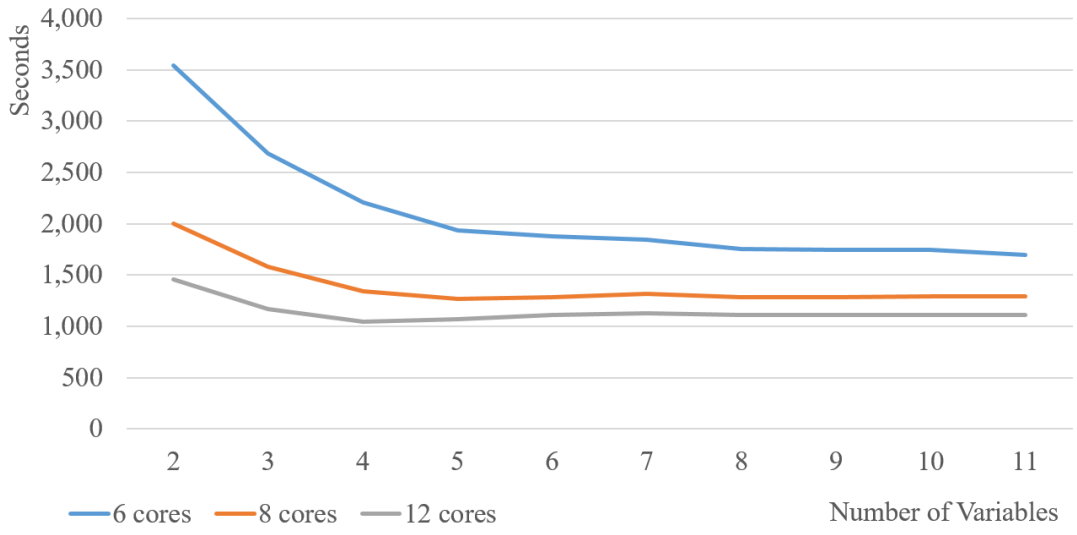

Fig. 5.1: Number of variables

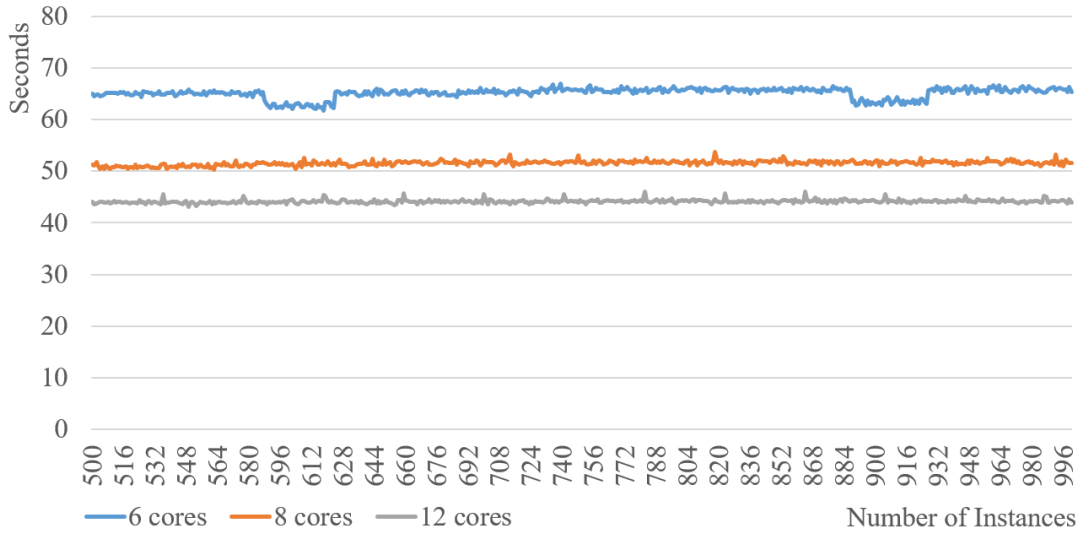

Fig. 5.2: Number of instances

generated with the exact same characteristics: Potentials with 100 Focal Elements, each Focal Element with 100 Configurations, and each Configuration with 2 Elements (one for each Variable). The difference will be that there are more options for selecting the 2 Elements in the Configurations, but this has no impact when traversing the Graph for the Combination Operation.

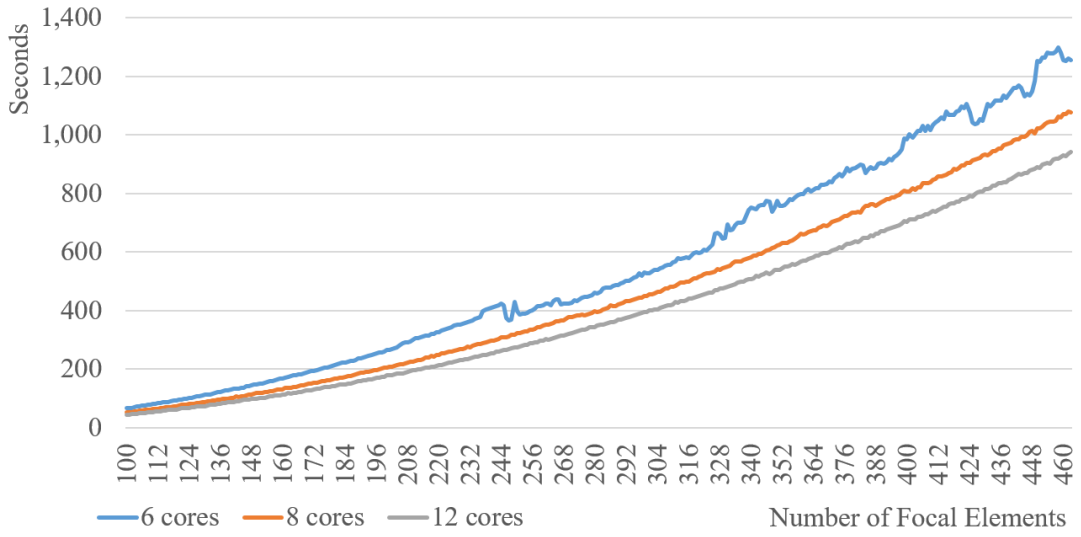

Fig. 5.3: Number of focal elements

Figure 5.3 presents the execution times when the number of Focal Elements is increased. We see that indeed it leads to an increased execution time. This is because this has a direct impact on the size of the graph generated for each Potential, and as we saw in Chapter 4, this is the biggest factor for the execution time.

Increasing the number of Configurations also leads to an increased execution time, as seen in Figure 5.4. As with the Focal Elements, the number of Configurations have a direct impact on the size of the graph of the Potential.

5.2 Usefulness Validation

To test the usefulness of the proposed approaches in detecting riots in CCTV scenes, we carried out experiments in an actual CCTV dataset with the presence of riots. The tests consisted of using the approach proposed in Chapter 3 with the Framework proposed in Chapter 4. We will now describe in detail the dataset used as well as the setup used for the experiments.

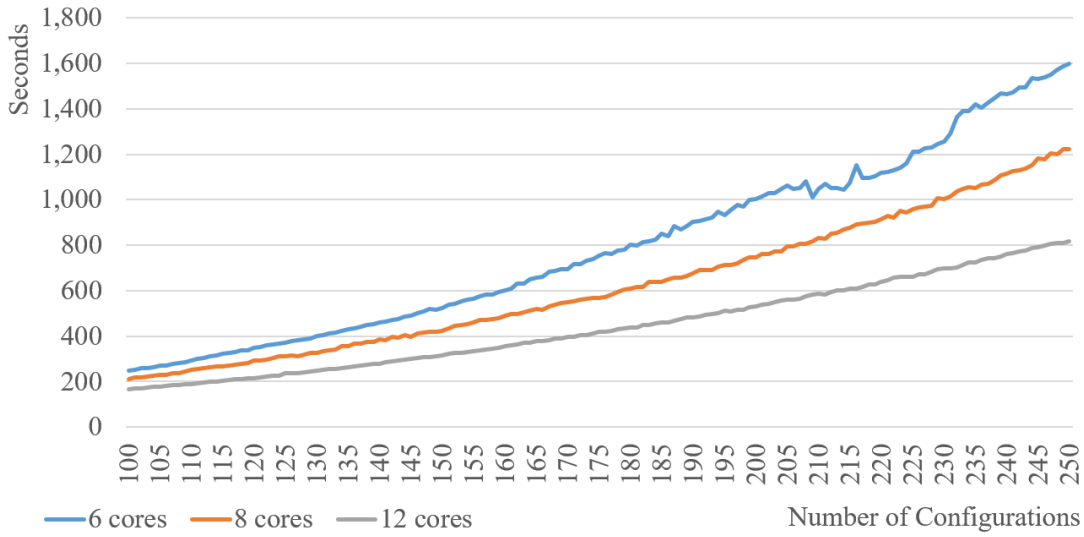

Fig. 5.4: Number of configurations

5.2.1 Riot Detection Dataset

To the best of our knowledge, there is no dataset in the literature for the task of Riot detection. For this reason, we created our own Riot detection dataset. The dataset consists of 148 videos collected by the London MET police after the London Riots in 2011 as evidence for the proceeding police and legal cases. The dataset consists of a total of 1,562,516 frames, taken from different CCTV cameras in public areas around the London borough of Hackney. The videos where manually annotated for the presence of different concepts in each frame. The ground truth annotation was performed using the ViperGT annotation tool [65]. Six concepts in total where annotated:

- **Running**: Whenever there was someone running in the scene.
- **Face Covered**: This included people with the face covered with the intention of concealing their identity with what might include scarves, bandannas, hooded sweatshirts, and similar.
- **Crowd**: For the purposes of this task, a crowd is defined as a group of 5 or more people.

- **Fire**: Presence of fire in the scene.
- **Vandalism**: This is defined as the deliberate destruction or damage of public or private property.
- **Riot**: A riot could be defined as a crowd committing acts of vandalism.

Only if the concept is present in the frame, at least partially, it is annotated as being present. For example, if the camera is recording a riot but it zooms in to a small group of people within the riot, then the riot concept is no longer present as there is no longer a crowd in the scene.

An example of a frame from the dataset can be seen in Figure 5.5. Faces have been blurred to protect the privacy of the people involved. On this frame we can see an example of a frame that would be positive for "Vandalism", "Face Covered", and "Riot". More information on the annotations can be found in Appendix B.

The videos where acquired through the MET Police as part of the ongoing collaboration in the LASIE project [66] with the MMV Lab at QMUL. Appropriate data protection measures have been implemented and the videos have only been used for academic purposes.

This also constitutes our final contribution, as we are making the ground truth annotations for this dataset publicly available. It is available to download at `http://www.eecs.qmul.ac.uk/~chps3/riotDetectionDataset.zip`. Due to privacy and ethical reasons, only the ground truth annotations can be share at this point.

5.2.2 Experimental Setup

To be able to use the TBM Reasoner for the Semantic Web, we must first create (or pick an existing) ontology of the domain we are working with. In our case, we created a CCTV ontology with the relevant concepts. A graphical representation of the ontology

Fig. 5.5: Example of "Vandalism" and "Face Covered" from the dataset

can be seen in figure 5.6.

The ontology URI is `http://mmv.eecs.qmul.ac.uk/ontologies/riotDetection`. It is available to download at `http://www.eecs.qmul.ac.uk/~chps3/riotDetection.owl`. The concepts present in the ontology are (were `riot:` is the prefix for the namespace):

- `riot:MediaItem` This concept represents a single CCTV video.
- `riot:Frame` This is a single frame of a CCTV video.
- `riot:Concept` These are the different concepts present in our domain. Each of the subclasess have two instances, one to indicate the presence of the concept and one to indicate the absence of it. For example, the concept `riot:Running` has the instances `riot:running` and `riot:no_running`.
- `TBM:Potential` This is a reference to the concept in the `TBM` ontology.

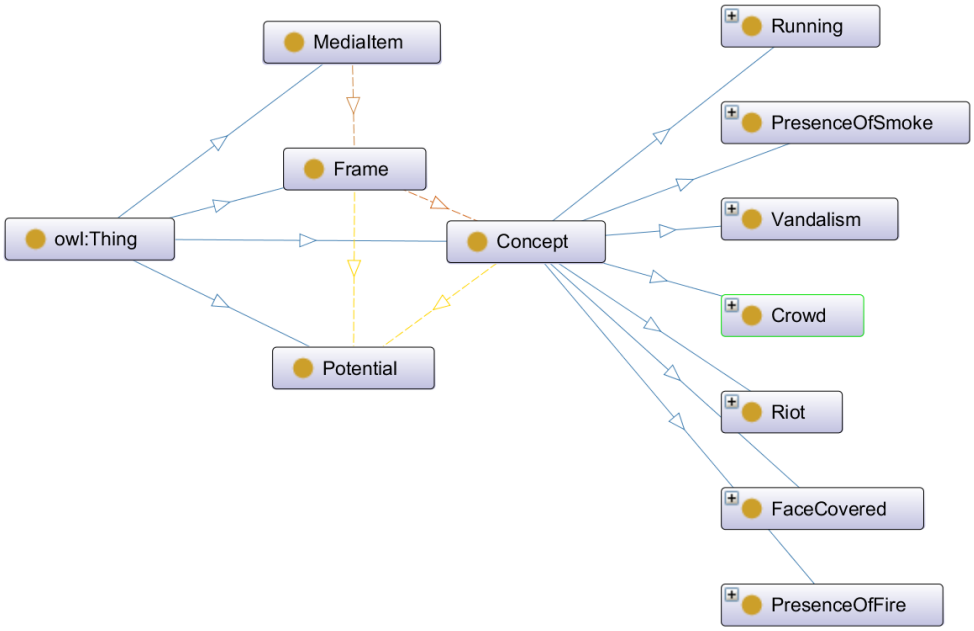

Fig. 5.6: Riot Detection Ontology

The properties in the ontology are:

- `riot:hasFrameNumber` This is the only Data Property. Its domain is `riot:Frame`, and its range is `xsd:unsignedLong`. This represents the frame number of a given Frame.

- `riot:hasFrame` The domain is `riot:MediaItem` and the range `riot:Frame`. This links the CCTV videos and its Frames.

- `riot:hasGlobalPotential` The domain is `riot:Frame` or `riot:Concept` and the range is `TBM:Potential`. This concept links the Concepts (for the training phase) or the Frames (for the reasoning phase) with their respective global potentials, as explained in Chapter 3.

Following the approach presented in Chapter 3, we randomly selected roughly 30% of the annotations (474,035) to train the model. Although this 30/70 split was arbitrary, it

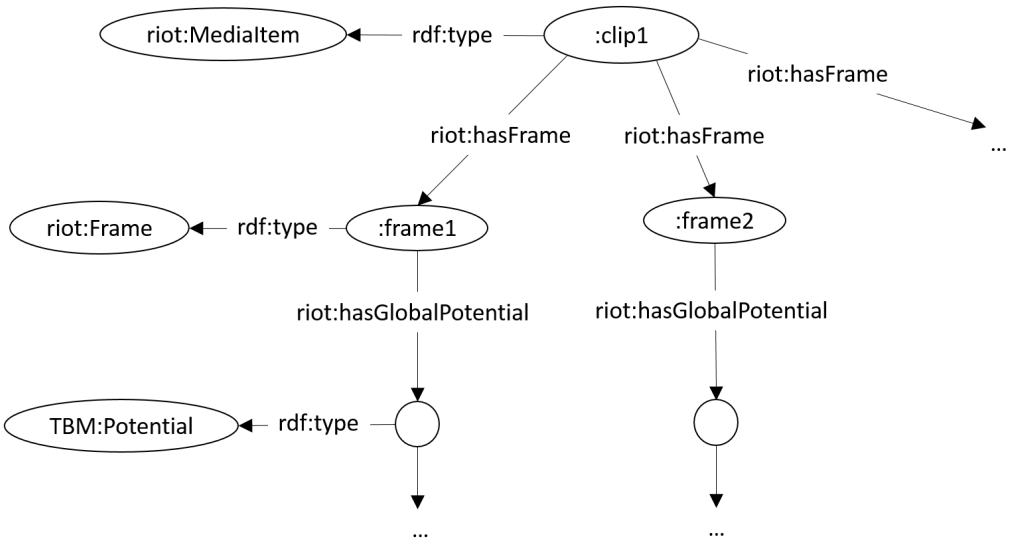

Fig. 5.7: Example resulting Riot graph

was done based on previous experiments with Machine Learning. The training consists of first calculating the probabilities of each pair of concepts. Then, for each concept, the global potential is calculated by increasingly combining every other concept.

Once the global potentials for each concept have been computed, the actual population of the Knowledge Base takes place. For the remaining 70% of frames, for each concept present in the frame, the global potential of that concept is combined with the global potential of that frame, starting with the potential for total ignorance (Ω). At the end of this step, we end up with a graph similar to the one presented in 5.7. When populating the Knowledge Base, the concept of Riot is left out, expecting its value to be populated from the probabilities of the other concepts. We then compare the belief of Riot with the actual presence of Riot from the GT.

5.2.3 Performance Metrics

At the end of the process, we have a belief from 0 to 1 that the given frame has Riot, given the presence (or absence) of the other concepts. To evaluate the effectiveness of

our approach, we convert this problem to a classification problem, where beliefs above a certain threshold are considered as belonging to the riot class, and vice versa.

For this, we use the precision/recall metric. Precision is the ratio of true positives in the result. A precision of 1.0 would mean that all of the classified elements are true positives, but there could be other misclassified documents.

On the other hand, recall is the ratio of false negatives in the classified items. A recall of 1.0 means that all of the relevant elements where correctly classified, but other non-relevant documents could have also been retrieved.

There is often an inverse relation between these two measures, when you increase one at the cost of reducing the other. In our evaluation, we could increase the precision by increasing the threshold (thus classifying less elements as riot), but this leads to a reduced recall by leaving some correct elements out.

By contrast, recall could be increased by reducing the threshold, including more elements in the classification, but this would include also other non-relevant elements (reducing precision).

This measures are commonly used together to measure the overall performance of an information retrieval or classification approach, where either values for one measure are compared for a fixed level at the other measure (e.g. precision at a recall level of 0.75). For our experiments, we raised the value of the threshold by 0.1 each time, starting form 0.40.

However, the value of this approach resides more in the ability to give a value of the belief or confidence that a given frame has riot. The evaluation will consider this as well.

Note that given the large nature of the data being used, it is evident that it can not fit on the memory of a single computer. Because of this, we used a TDB backed storage, which would limit the size of the data only to the size of the computer's disk. We also limited the processing to every 12 frames (every half a second in a standard 24fps video,

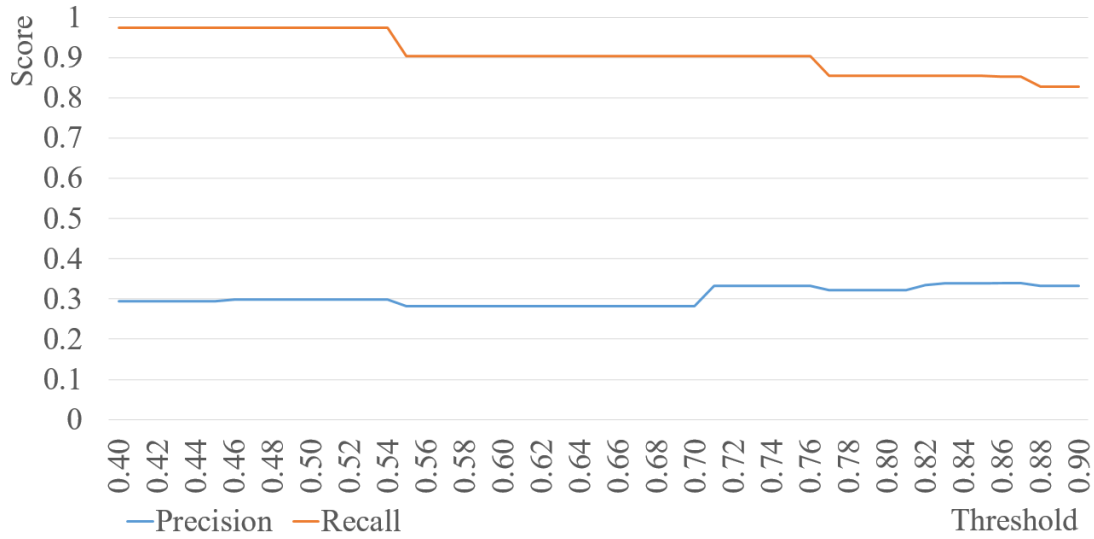

Fig. 5.8: Precision-Recall

or every second in a 12fps video).

5.2.4 Results

Figure 5.8 presents the results of the evaluation. As a starting point, we can see that even at low threshold levels, the recall is very good. This means that a majority of the frames that are classified in the ground truth as containing Riots, are correctly identified.

However we can also see that the precision is not very good. This means that the approach produces a lot of false positives. This is most likely due to the high presence of other concepts that highly entail the presence or Riot in the frame. And because the strength of this approach is the construction of a belief given the presence of other concepts, there will always be a certain belief of the presence of the concepts. Another possibility is that since we are using the closed world assumption, all of the masses get normalised in case of a conflict. This means that it is possible that some of the belief masses are not that high, but since they get normalised, their value gets increased so the mass of the potential adds up to 1 again.

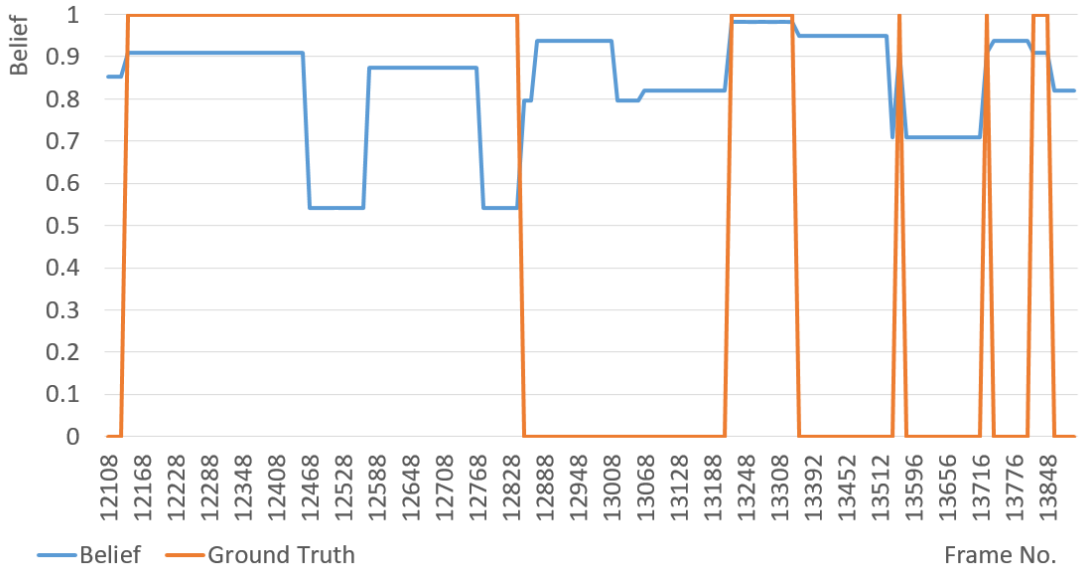

Fig. 5.9: Belief of Riot in a segment with concepts which entail Riot

Fig. 5.10: Frame of a segment with concepts which entail Riot

For this, let us look into more detail how the framework behaves when there are concepts which highly entail the presence of Riot, and when there are not.

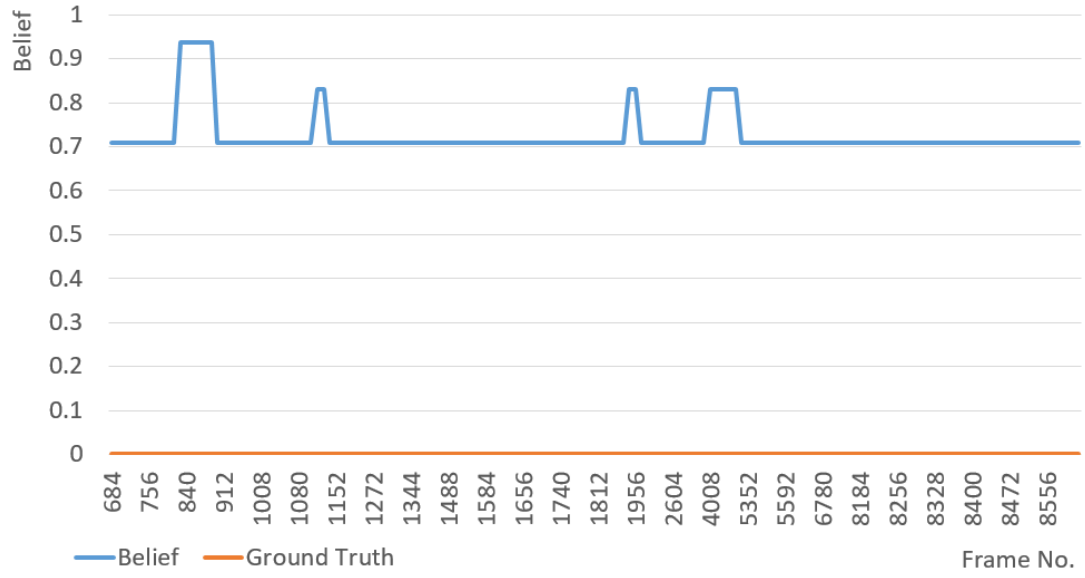

Fig. 5.11: Belief of Riot in a segment without concepts which entail Riot

Figure 5.9 presents the belief of Riot, against the actual annotated value from the ground truth in a video segment which has the presence of crowd, running, and people with the face covered. Concepts which highly entail the presence of Riot as seen in the probability matrix presented in Chapter 3. A screen shot of the video can be seen on Figure 5.10. Figure 5.11 on the other hand, presents the belief of Riot against the actual value from the ground truth in a segment without the presence of concepts which highly entail the presence of Riots. A screen shot of the video can be seen on Figure 5.12.

On these charts we can see two things: the first one is that in effect when there are no concepts that highly entail the Riot concept, there is some belief in the system about the presence of Riot, but it is not as high as when there are other concepts that highly entail Riot. The second thing is that there are certain spikes in the belief that there is riot when more concepts that highly entail it come into the scene. This results allow us to say that this is a valid approach worth exploring further.

Fig. 5.12: Frame of a segment without concepts which entail Riot

5.3 Concluding Summary

On this Chapter, we have evaluated the proposed framework in two ways: its ability to work with large datasets and the correctness of the results it produces. Both tests produce promising results, but show that there is still room for improvement.

On the performance evaluation we can see that the system does not change its runtime significantly when some parameters are varied, but it does run at very high times with very large inputs for others. However, it is also clear that it scales very well with more resources. This last part is very promising as computers become more powerful but it also opens the door to try and apply other parallel approaches to the problem. Apache Spark, Map-Reduce, and General Processing Graphical Processing Unit (GPGPU) computing are good candidates for this, as all of them distribute the processing over many compute units.

A data set consisting of more that 1.5 million frames with annotations for 6 concepts

related to riot detection was developed for the Theoretical Validation. This data set is an important contribution to the community and we looking forward on how the research community will use it .

The Theoretical validation shows that the combination of the proposed approaches is able to detect accurately the presence of riots, however at the cost of a lot of false positives. Future work on this aspect should be focused on lowering the false positive count, or using the belief mass produced by the framework in a more meaningful way. For example, although a frame might not contain riot directly, a high belief of riot might point to a high risk of there being a riot temporally or spatially near.

Chapter 6

Conclusions

We have developed and implemented a framework to infer the presence of Riots in CCTV recordings. For this we implemented a TBM semantic reasoner. This has the advantages of being large-scale ready, benefiting from an active research community, and interoperability with other semantic web data sources.

Our framework deals with imprecise and uncertain information using the TBM. We apply this Framework to the main task of this work, which is to infer the presence of Riot in CCTV recordings. Next, the contributions are listed in the order of the Chapters they were introduced.

In Chapter 2, we presented a review of different domains, all relevant for this research. Namely: The Semantic Web and its surveillance applications, previous attempts at representing and reasoning on the Semantic Web with imprecise and uncertain information, the Transferable Belief Model and its previous implementations. This served as solid base on which to lay the following chapters, which presented the main body of research.

Chapter 3 proposes a method for combining traditional Bayesian *a priory* probabilities with the TBM. This allows us to take advantage of not only the case when *a priori* probabilities are available, but also when information is coming from multiple hetero-

geneous and potentially unreliable sensors. It is done in two stages, the first one is the training where the probabilities are computed and the global Potentials for each concept constructed. Once this has been done, we can apply this knowledge to available data. This could be from live sensors or historical data. Every time we learn a concept that is present in the data, we include the knowledge of the presence not only of that concept but also all the concepts it entails with its respective probabilities. We illustrated our contribution with the riot detection case, where *a priori* probabilities for multiple concepts are available. These probabilities are combined using the TBM and then the model queried about the knowledge of the presence of riots in a given instant.

Three of the contributions of this this thesis are presented in Chapter 4, namely the TBM Ontology, the TBM Reasoner, and the parallel heuristics of the TBM operations. They form the bulk of the TBM Semantic Web framework presented in this thesis. The Framework is developed in Java and used the Jena API for Ontologies. The operations of the framework can be accessed through the Java API or through SPARQL queries. This framework allows us to represent and reason with imprecise and uncertain information on the semantic web. The parallel heuristics are crucial to this, as they allow the system to be scalable (use more computing power as it becomes available) and to work on inputs of thousands of records.

Chapter 5, presents the evaluation of the Framework together with the last contribution, which is the ground truth annotations of actual CCTV recordings from the 2011 London riots. The annotations consists of 1,562,516 frames on 148 videos and can be downloaded freely from `http://www.eecs.qmul.ac.uk/~chps3/riotDetection.owl`. The evaluation consisted in a performance and scalability test, to test the feasibility of applying the framework on large datasets, as well as a usefulness validation of the approach presented in Chapter 3 using the proposed framework applied to the task of Riot detection. The evaluation shows two things: first, that the TBM reasoner for the semantic web works. Second, that the proposed approach is able to detect Riots in CCTV recordings.

Our performance tests confirm the high computational cost of the combination operation when indexed collections are used to represent the Focal Elements, and that the number of Focal Elements and Configurations have the biggest impact on the computation time.

Our approach does have some long computation times. For example, for 460 Focal Elements on a 12 cores machine, the combination took about 900 seconds or 15 minutes. Although regular problems are not expected to be that big [24] multimedia problems such as the ones proposed here will, which is why we are looking to further optimise our approach.

Our results also show that the TBM, and in particular our approach, is scalable. We see in our results that for equivalent sets of inputs, more cores resulted in less execution time. For example on the Number of Configurations test, there was an improvement of around half of the execution time when going from 8 cores to 12.

Comparison to other implementations of the TBM is not feasible, as complexity is highly dependant on the data structures used in each implementation. As stated before, bitset representation of the Focal Elements provides a much reduced computation time, at the expense of a much increased memory usage. Additionally, other implementations in the literature, like in [1] and [24] use different technologies (for example C++ and Lisp instead of Java).

Finally, on the usefulness validation, our tests show that the system is good at detecting frames with riot, but at the cost of a lot of false positives. This could be explained by the fact that this dataset is particularly tailored for riots and that the output of the system is not a class, but a score of the belief of the concept in the frame. This means that even for frames that do not have riots, there will always be even a small belief that there is riots on the scene. Another possible explanation is that since we are working under the Closed World assumption, in the case of conflict when combining two potentials, the masses are normalised to add up to 1 again, which increases the beliefs.

One immediate improvement to the current system would be the reduction of the false positives rate. As seen in the evaluation, precision was very low. Future efforts have to be put to try to minimise the precision, trying not to affect the recall.

We would also like to combine different forms of reasoning with the TBM reasoning. This could lead to interesting applications where for example an OWL reasoner provides a model with inferred tuples to a TBM model with class inference. Or the opposite, a TBM model providing a model for an OWL model to perform inference on. This could also be a potential method of reducing the false positive rate.

Implementation of the regrouping operation is also a possible area of improvement. After two potentials have been combined, it is possible for two or more generated Focal Elements to have the same Configurations. In such cases, the masses can be added which would allow to keep just one Focal Element. In our current implementation, the Focal Elements are not regrouped. Although this does not change the value of the masses, it would be ideal to implement the Regrouping Operation, which would reduce the space requirements of the Framework, in exchange of added computational complexity.

Another possible improvement is that in our current approach, whenever there is conflicting masses, the mass of the remaining Focal Elements is normalised so they add up to 1 again. This is because we follow the closed world assumption where the truth must always be within the system. However, the Semantic Web works under the Open World paradigm, where if an assertion is not present in the system, it does not mean that it does not exists, simply that it is not known yet. Given that the TBM already has a mechanism to detecting the conflict in a system, it would be desirable to use that conflict in a meaningful way. It could be used for example to detect anomalies in a system. In our approach it could happen that two concepts that in the training data where never together, but are detected as occurring at the same time later. One interesting approach would be something similar to what has been done in [53], where the conflicting mass is assigned to a special focal element. However a mechanism would have to be defined to update the system when new information that removes the conflict becomes available.

Due to the nature of number representation in binary systems such as digital computers, rounding errors are inevitable when dealing with real numbers. Our approach introduces some very small rounding errors in the calculation of the masses which do not affect greatly the end result. However, it would be desirable to get rid of the rounding errors altogether. Solutions for this could be proposed in the future.

Finally the combination operation, and in particular its performance, can be further optimised. Several alternatives are being explored:

Map-Reduce : It is a programming model for massively parallel systems which can consist of multiple compute nodes. The core principles are the *map* operation, and the *reduce* operation. The *map* operation splits the data into different chunks by a defined key, the *reduce* operation performs some computation on the split data. As an initial idea to apply the combination operation using *map-reduce*, the *map* operation would split the different pair of Configurations possible from the two Potentials being combined. The *reduce* operation would then perform the actual intersection of the two Configurations.

Apache Spark : It is a framework which provides APIs for cluster computing. Spark can be seen as a superset of *map-reduce* where not just those two operations are supported, and data is only written to disk when is absolutely necessary. This allows to have jobs with much more stages and sharing the same data on memory (when possible). A bespoke algorithm for the Combination operation would be developed with Apache Spark as the back-end.

General Processing Graphical Processing Unit (GPGPU) Computing : With the advent of powerful graphics cards, came the possibility of programming them to perform not just graphics tasks, but also general purpose computing. The Combination operation has certain characteristics that make it ideal for implementation in GPUs: the loops share a lot of data, and it is the same operation applied over and over again (fulfill-

ing the SIMT paradigm of GPGPU computing). The actual operation inside the loops could be seen as an intersection between the Configurations of the Focal Elements and in fact, [67] already proposes an intersection algorithm for GPUs. Under this paradigm, all of the combinations of Focal Elements would be uploaded to the GPU's memory, each block would deal with the combination of different Focal Elements, and the threads in a block would perform the actual intersection of the elements in configurations.

List of Publications

Academic Papers

C. Pantoja, E. Izquierdo, "Design and Evaluation of a Semantic Web Reasoner based on the Transferable Belief Model", *Multimedia Tools and Applications - Special Issue on "Content Based Multimedia Indexing"*, 2016 (submitted)

C. Pantoja, E. Izquierdo, "A Novel Architecture of Semantic Web Reasoner Based on Transferable Belief Model", *14th International Workshop on Content-Based Multimedia Indexing (CBMI)* , pp. 1-6 , 2016

C. Pantoja, E. A. Ciapetti, C. Massari, M. Tarantelli, "Action recognition in surveillance videos using semantic web rules", *6th International Conference on Imaging for Crime Prevention and Detection (ICDP-15)* , pp. 1-6 , 2015

C. Pantoja, E. Izquierdo, "MediaEval 2014 Visual Privacy Task: De-identification and Re-identification of Subjects in CCTV", *Working Notes Proceedings of the MediaEval 2014 Workshop* , Vol. 1263 , pp. 1-2 , 2014

C. Pantoja, V. Fernandez Arguedas, E. Izquierdo, "MediaEval 2013 Visual Privacy Task: Pixel Based Anonymisation Technique", *Proceedings of the MediaEval 2013 Multimedia Benchmark Workshop* , Vol. 1043 , pp. 1-2 , 2013

C. Pantoja, V. Fernandez Arguedas, E. Izquierdo, "Anonymization and De-identification of Personal Surveillance Visual Information: A Review", *Proceedings of the 5th Latin-American Conference on Networked and Electronic Media (LACNEM 2013)* , pp. 1-6 , 2013

Posters

C. Pantoja, E. Izquierdo, "Transferable Belief Model for the Semantic Web", *6th Latin American Conference on Networked and Electronic Media (LACNEM2015)*, 2015

References

[1] N. Burrus and D. Lesagne, *Theory of Evidence*, 2003.

[2] A. Bellenger, S. Gatepaille, H. Abdulrab, and J.-P. Kotowicz, "An evidential approach for modeling and reasoning on uncertainty in semantic applications," in *Proceedings of the 7th International Conference on Uncertainty Reasoning for the Semantic Web - Volume 778*, ser. URSW'11. Aachen, Germany, Germany: CEUR-WS.org, 2011, pp. 27–38. [Online]. Available: http://dl.acm.org/citation.cfm?id=2887702.2887705

[3] Apache Software Fundation, "Jena ontology api introduction," https://jena.apache.org/documentation/ontology/, Dec. 2015.

[4] "Asociation of british insurers news release," https://web.archive.org/web/20120301213043/http://www.abi.org.uk/Media/Releases/2011/08/UK__Insurance_Industry_welcomes_Prime_Ministers_Compensation_Scheme_announcement_and_pledges_help_to_make_the_scheme_work_.aspx, Aug. 2011.

[5] R. J. Rosen, "London riots, big brother watches: Cctv cameras blanket the uk," http://www.theatlantic.com/technology/archive/2011/08/london-riots-big-brother-watches-cctv-cameras-blanket-the-uk/243356/, Aug. 2011.

[6] S. Laville, "London riots: 450 detectives in hunt for looters," https://www.theguardian.com/uk/2011/aug/09/london-riots-detectives-police, Aug. 2011.

[7] B. News, "London riots: Most wanted suspect cctv images released," http://www.bbc.co.uk/news/uk-england-london-16171972, Dec. 2011.

[8] T. C. A. Service, "Cctv and london riots," http://www.cctv-information.co.uk/i/

CCTV_and_London_Riots, Aug. 2015.

[9] W3C Incubator Group, "Uncertainty reasoning for the world wide web," http://www.w3.org/2005/Incubator/urw3/XGR-urw3-20080331/, Apr. 2015.

[10] M. Smithson, *Ignorance and uncertainty: emerging paradigms*. Springer Science & Business Media, 2012.

[11] P. Smets, "Varieties of ignorance and the need for well-founded theories," *Information Sciences*, vol. 57, pp. 135–144, 1991.

[12] S. Han, B. Koo, and W. Stechele, "Subjective logic based approach to modeling default reasoning for visual surveillance," in *Semantic Computing (ICSC), 2010 IEEE Fourth International Conference on*, 2010, pp. 112–119.

[13] J. Gomez-Romero, M. A. Patricio, J. Garcia, and J. M. Molina, "Ontology-based context representation and reasoning for object tracking and scene interpretation in video," *Expert Systems with Applications*, vol. 38, no. 6, pp. 7494 – 7510, 2011. [Online]. Available: http://www.sciencedirect.com/science/article/pii/S0957417410014818

[14] T. Berners-Lee, J. Hendler, O. Lassila *et al.*, "The semantic web," *Scientific american*, vol. 284, no. 5, pp. 28–37, 2001.

[15] W. Recommendation, "Rdf 1.1 concepts and abstract syntax," https://www.w3.org/TR/rdf11-concepts/, Feb. 2014.

[16] W3C, "Owl 2 web ontology language profiles (second edition)," https://www.w3.org/TR/owl2-profiles/, Dec. 2012.

[17] Holger Knublauch and Kingsley Idehen and James A. Hendler, "Sparql inferencing notation (spin) - w3c member submission," https://www.w3.org/Submission/2011/02/, Feb. 2011.

[18] A. Oltramari and C. Lebiere, "Using ontologies in a cognitive-grounded system: Automatic action recognition in video surveillance," 2012. [Online]. Available: http://stids.c4i.gmu.edu/papers/STIDSPapers/STIDS2012_T02_OltramariLebiere_CognitiveGroundedSystem.pdf

[19] J. C. SanMiguel and J. M. Martinez, "A semantic-guided and self-configurable framework for video analysis," *Machine Vision and Applications*, vol. 24, no. 3, pp.

493–512, 2013. [Online]. Available: http://dx.doi.org/10.1007/s00138-011-0397-x

[20] C. Pantoja, A. Ciapetti, C. Massari, and M. Tarantelli, "Action recognition in surveillance videos using semantic web rules," in *6th International Conference on Imaging for Crime Prevention and Detection (ICDP-15)*, July 2015, pp. 1–6.

[21] A. P. Dempster, "Upper and lower probabilities induced by a multivalued mapping," *The Annals of Mathematical Statistics*, vol. 38, no. 2, pp. 325–339, 1967. [Online]. Available: http://www.jstor.org/stable/2239146

[22] G. Shafer *et al.*, *A mathematical theory of evidence*. Princeton university press Princeton, 1976, vol. 1.

[23] P. Smets and R. Kennes, "The transferable belief model," *Artificial intelligence*, vol. 66, no. 2, pp. 191–234, 1994.

[24] R. Haenni and N. Lehmann, "Implementing belief function computations," *International Journal of Intelligent Systems*, vol. 18, no. 1, pp. 31–49, 2003.

[25] "Evidencce4j project website," https://github.com/IGNF/evidence4j, Jan. 2014.

[26] J. Klein, C. Lecomte, and P. Miche, "Preceding car tracking using belief functions and a particle filter," in *Pattern Recognition, 2008. ICPR 2008. 19th International Conference on*, Dec 2008, pp. 1–4.

[27] R. Munoz-Salinas, R. Medina-Carnicer, F. Madrid-Cuevas, and A. Carmona-Poyato, "Multi-camera people tracking using evidential filters," *International Journal of Approximate Reasoning*, vol. 50, no. 5, pp. 732 – 749, 2009. [Online]. Available: http://www.sciencedirect.com/science/article/pii/S0888613X09000346

[28] Z. Ding, Y. Peng, and R. Pan, "Bayesowl: Uncertainty modeling in semantic web ontologies," in *Soft Computing in Ontologies and Semantic Web*, ser. Studies in Fuzziness and Soft Computing, Z. Ma, Ed. Springer Berlin Heidelberg, 2006, vol. 204, pp. 3–29. [Online]. Available: http://dx.doi.org/10.1007/978-3-540-33473-6_1

[29] P. C. G. da Costa, K. B. Laskey, and K. J. Laskey, "Pr-owl: A bayesian ontology language for the semantic web," in *ISWC-URSW*, 2005, pp. 23–33.

[30] P. C. de Oliveira, "Probabilistic Reasoning in the Semantic Web using Markov Logic," Master's thesis, University of Coimbra, 2009.

[31] Y. Fukushige, "Representing probabilistic knowledge in the semantic web," http:

//www.w3.org/2004/09/13-Yoshio/PositionPaper.html, Apr. 2015.

[32] F. Bobillo, R. Carvalho, D. Ceolin, P. C. G. da Costa, C. d'Amato, N. Fanizzi, K. B. Laskey, K. J. Laskey, T. Lukasiewicz, T. Martin, M. Nickles, and M. Pool, Eds., *Proceedings of the 12th International Workshop on Uncertainty Reasoning for the Semantic Web (URSW 2016)*. CEUR-WS.org, Oct. 2016. [Online]. Available: http://ceur-ws.org/Vol-1665/

[33] L. Predoiu and H. Stuckenschmidt, "Probabilistic models for the semantic web: A survey," *The Semantic Web for Knowledge and Data Management: Technologies and Practices. Information Science Reference, Hershey, PA, USA*, 2008.

[34] S. Han, A. Hutter, and W. Stechele, "Toward contextual forensic retrieval for visual surveillance: Challenges and an architectural approach," in *Image Analysis for Multimedia Interactive Services (WIAMIS), 2009 10th International Workshop on*, 2009, pp. 201–204.

[35] S. Han, B. Koo, A. Hutter, and W. Stechele, "Forensic reasoning upon pre-obtained surveillance metadata using uncertain spatio-temporal rules and subjective logic," in *Image Analysis for Multimedia Interactive Services (WIAMIS), 2010 11th International Workshop on*, 2010, pp. 1–4. [Online]. Available: http://ieeexplore.ieee.org/xpls/abs_all.jsp?arnumber=5617680

[36] G. Riley, C. Culbert, and F. Lopez, "C language integrated production system," 1989.

[37] D. Ceolin, A. Nottamkandath, and W. Fokkink, "Subjective logic extensions for the semantic web," in *8th International Workshop on Uncertainty Reasoning for the Semantic Web*, 2012.

[38] A. F. K. S. Y. T. Jhonatan Garcia, Jeff Z. Pan and F. Cerutti., "Handling uncertainty: An extension of DL-Lite with Subjective Logic," in *Proc. of 28th International Workshop on Description Logics (DL 2015).*, 2015.

[39] M. Sensoy, J. Z. Pan, A. Fokoue, M. Srivatsa, and F. Meneguzzi, "Using subjective logic to handle uncertainty and conflicts," in *2012 IEEE 11th International Conference on Trust, Security and Privacy in Computing and Communications*. IEEE, 2012, pp. 1323–1326.

[40] F. Bobillo and U. Straccia, "Fuzzy ontology representation using owl 2," *Int. J. Approx. Reasoning*, vol. 52, no. 7, pp. 1073–1094, Oct. 2011. [Online]. Available: http://dx.doi.org/10.1016/j.ijar.2011.05.003

[41] G. Stoilos, G. Stamou, V. Tzouvaras, J. Z. Pan, and I. Horrock, "A Fuzzy Description Logic for Multimedia Knowledge Representation," in *Proc. of the International Workshop on Multimedia and the Semantic Web*, 2005.

[42] G. Stoilos, G. Stamou, V. Tzouvaras, J. Z. Pan, and I. Horrocks, "The Fuzzy Description Logic f-SHIN," in *Proc. of the International Workshop on Uncertainty Reasoning for the Semantic Web*, 2005.

[43] G. Stoilos, G. Stamou, and J. Z. Pan, "Handling imprecise knowledge with fuzzy description logic," in *Proc. 2006 Internat. Workshop on Description Logics (DL 2006)*, 2006, pp. 119–126.

[44] G. Stoilos, U. Straccia, G. Stamou, and J. Z. Pan, "General concept inclusions in fuzzy description logics," *FRONTIERS IN ARTIFICIAL INTELLIGENCE AND APPLICATIONS*, vol. 141, p. 457, 2006.

[45] G. Stoilos, G. Stamou, J. Z. Pan, V. Tzouvaras, and I. Horrocks, "Reasoning with Very Expressive Fuzzy Description Logics," *Journal of Artificial Intelligence Research*, to appear.

[46] G. Stoilos, G. Stamou, and J. Z. Pan, "Classifying fuzzy subsumption in fuzzy-el," in *In Description Logics*, 2008.

[47] G. Stoilos, G. Stamou, and J. Z. Pan., "Fuzzy Extensions of OWL: Logical Properties and Reduction to Fuzzy Description Logics," in *International Journal of Approximate Reasoning*, 2010.

[48] J. Z. Pan, G. Stamou, G. Stoilos, and E. Thomas, "Expressive Querying over Fuzzy DL-Lite Ontologies," in *Proc. of 2007 International Workshop on Description Logics (DL2007)*, 2007.

[49] J. Z. Pan, G. Stamou, G. Stoilos, S. Taylor, and E. Thomas, "Scalable Querying Services over Fuzzy Ontologies," in *the Proc. of the 17th International World Wide Web Conference (WWW2008)*, 2008.

[50] C. V. Damasio, J. Z. Pan, G. Stoilos, and U. Straccia, "Representing Uncertainty

in RuleML," *Fundamenta Informaticae*, 2007, to appear.

[51] J. Z. Pan, G. Stoilos, G. Stamou, V. Tzouvaras, and I. Horrocks, "f-swrl: a fuzzy extension of swrl," in *Journal on Data Semantics VI*. Springer, 2006, pp. 28–46.

[52] A. Nikolov, V. Uren, E. Motta, and A. de Roeck, *Using the Dempster-Shafer Theory of Evidence to Resolve ABox Inconsistencies*. Berlin, Heidelberg: Springer Berlin Heidelberg, 2008, pp. 143–160. [Online]. Available: http://dx.doi.org/10.1007/978-3-540-89765-1_9

[53] G. Rizzo, N. Fanizzi, C. d'Amato, and F. Esposito, "Prediction of class and property assertions on owl ontologies through evidence combination," in *Proceedings of the International Conference on Web Intelligence, Mining and Semantics*, ser. WIMS '11. New York, NY, USA: ACM, 2011, pp. 45:1–45:9. [Online]. Available: http://doi.acm.org/10.1145/1988688.1988741

[54] G. Rizzo, C. d'Amato, N. Fanizzi, and F. Esposito, *Assertion Prediction with Ontologies through Evidence Combination*. Berlin, Heidelberg: Springer Berlin Heidelberg, 2013, pp. 282–299. [Online]. Available: http://dx.doi.org/10.1007/978-3-642-35975-0_15

[55] G. Rizzo, C. d'Amato, N. Fanizzi, and F. Esposito, *Towards Evidence-Based Terminological Decision Trees*. Cham: Springer International Publishing, 2014, pp. 36–45. [Online]. Available: http://dx.doi.org/10.1007/978-3-319-08795-5_5

[56] M. Sensoy, A. Fokoue, J. Z. Pan, T. Norman, Y. Tang, N. Oren, and K. Sycara., "Reasoning about Uncertain Information and Conflict Resolution through Trust Revision," in *Proc. of the 12th International Conference on Autonomous Agents and Multiagent Systems (AAMAS2013).*, 2013.

[57] N. Lehmann, "Argumentation systems and belief functions," Ph.D. dissertation, Université de Fribourg, 2001.

[58] J. J. Carroll, I. Dickinson, C. Dollin, D. Reynolds, A. Seaborne, and K. Wilkinson, "Jena: Implementing the semantic web recommendations," in *Proceedings of the 13th International World Wide Web Conference on Alternate Track Papers &Amp; Posters*, ser. WWW Alt. '04. New York, NY, USA: ACM, 2004, pp. 74–83. [Online]. Available: http://doi.acm.org/10.1145/1013367.1013381

[59] R. Johnson, E. Gamma, J. Vlissides, and R. Helm, *Design Patterns: Elements of Reusable Object-Oriented Software*. Addison-Wesley, 1995. [Online]. Available: https://books.google.co.uk/books?id=iyIvGGp2550C

[60] R. D. Blumofe and C. E. Leiserson, "Scheduling multithreaded computations by work stealing," *J. ACM*, vol. 46, no. 5, pp. 720–748, Sep. 1999. [Online]. Available: http://doi.acm.org/10.1145/324133.324234

[61] R. Lee, "Scalability report on triple store applications," *Massachusetts institute of technology*, 2004.

[62] K. Rohloff, M. Dean, I. Emmons, D. Ryder, and J. Sumner, "An evaluation of triple-store technologies for large data stores," in *On the Move to Meaningful Internet Systems 2007: OTM 2007 Workshops*. Springer, 2007, pp. 1105–1114.

[63] B. Liu and B. Hu, "An evaluation of rdf storage systems for large data applications," in *Semantics, Knowledge and Grid, 2005. SKG'05. First International Conference on*. IEEE, 2005, pp. 59–59.

[64] E. Minack, W. Siberski, and W. Nejdl, "Benchmarking fulltext search performance of rdf stores," in *The Semantic Web: Research and Applications*. Springer, 2009, pp. 81–95.

[65] "Vipergt website," http://viper-toolkit.sourceforge.net/, Jan. 2014.

[66] "Lasie project website," http://www.lasie-project.eu/, Oct. 2016.

[67] D. Wu, F. Zhang, N. Ao, F. Wang, X. Liu, and G. Wang, "A batched gpu algorithm for set intersection," in *Pervasive Systems, Algorithms, and Networks (ISPAN), 2009 10th International Symposium on*, Dec 2009, pp. 752–756.

Appendix A

Example Application

Now an example use case is presented with working application code. This is a simple application intended to showcase the functionalities of the semantic framework and how one might go about implementing a TBM-enabled application with our approach. We will base our example on the game of CLUEDO presented in [1]. We are tasked with finding the murderer, the weapon, and the room where the murder happened. The first player has a strong subjective belief of 80% that the murderer is "Colonel Mustard" and the murder happened in the "Kitchen". He doesn't have any information about the weapon. The second player believes also with a subjective 80% that the murderer is "Colonel Mustard" and the weapon is "Candlestick" but he is not sure about the room.

A.1 Problem Definition and Ontology

The first step towards implementing an application with our framework is to define the domain of the problem and encoding it in an OWL ontology. In this example, we created a simple ontology using the name space `http://mmv.eecs.qmul.ac.uk/CLUE.owl`. We used the Protègè Ontolgy Editor to create it, but any other mean of doing it is fine too as long as it's a standard OWL ontology. There are three classes: Murderer (with instances

"Colonel Mustard", "Miss Scarlett", and "Mrs. Peacock"), Room (with instances "Dining Room" and "Kitchen"), and Weapon (with instances "Dagger" and "Candle Stick"). A simplified version of the triples of the ontology in N3/turtle format is as follows:

```
@prefix : <http://mmv.eecs.qmul.ac.uk/CLUE.owl#> .
@prefix owl: <http://www.w3.org/2002/07/owl#> .
@prefix rdf: <http://www.w3.org/1999/02/22-rdf-syntax-ns#> .

:Murderer rdf:type owl:Class .
:Room     rdf:type owl:Class .
:Weapon   rdf:type owl:Class .

:ColonelMustard rdf:type :Murderer .
:MissScarlett   rdf:type :Murderer .
:MrsPeacock     rdf:type :Murderer .
:DiningRoom     rdf:type :Room .
:Kitchen        rdf:type :Room .
:CandleStick    rdf:type :Weapon .
:Dagger         rdf:type :Weapon .
```

A.2 Initialisation

We now move to the Java code used to model and reason with this problem. The first step is to initialise some variables that will be used thorughout the code. Namely: a string with the URL of the ontology, The Jena model that will store the tuples, and references to the resources that will be used later. Please note that not all of the instances of the variables are being referenced, as we only need those which will be used explicitly in the creation of the potentials, that is, `:ColonelMustard`, `:Kitchen`, and `:CandleStick`.

```
//Path on disk to the data files
```

```
String datasetPath = "./db";
//initialise the TDB storage
Dataset dataset = TDBFactory.createDataset(datasetPath);

//Create the model
TBMModel model = TBMModelFactory.createTBMModel(dataset.getDefaultModel());
//For a memory-backed storage use:
//TBMModel model = TBMModelFactory.createTBMModel(null);

//Namespace of the ontology
String ont = "http://mmv.eecs.qmul.ac.uk/CLUE.owl";
//read the ontology
model.read("file:CLUE.owl");

//instantiate variables
//murderer variable
Resource murderer = model.getResource(ont + "#Murderer"); //Type
Resource ColMustard = model.getResource(ont + "#ColonelMustard"); //Instance
//room variable
Resource room = model.getResource(ont + "#Room"); //Type
Resource Kitchen = model.getResource(ont + "#Kitchen"); //Instance
//weapon variable
Resource weapon = model.getResource(ont + "#Weapon"); //Type
Resource Candlestick = model.getResource(ont + "#CandleStick"); //Instance
```

Note that this will create a TDB-backed storage. For a memory-backed storage, create the model with the line: `TBMModel model = TBMModelFactory.createTBMModel(null);`

A.3 Players' Potentials

Here, the first player's knowledge is modelled. Please note that the first player's knowledge is only defined on the domain {`:Murderer, :Room`} and the second player is on {`:Murderer, :Weapon`}. The `TBMFocalElement::addAllConfigurations()` method is a helper method which adds the entire frame of discernment to the Focal Element. As we saw before this is done to model complete ignorance in the TBM. In this case, the focal elements and configurations are modelled as bnodes as it is not necessary to name them.

```
//first potential***********************
//instantiate p1 domain
TBMVarDomain d1 = model.createDomain();
d1.addVariable(murderer);
d1.addVariable(room);
//create first focal element
TBMFocalElement fe11 = model.createFocalElement();
fe11.setDomain(d1);
fe11.addConfiguration(ColMustard, Kitchen);
fe11.setMass(0.8);
//create second focal element
TBMFocalElement fe12 = model.createFocalElement();
fe12.setDomain(d1);
fe12.addAllConfigurations();
fe12.setMass(0.2);
//create potential with masses
TBMPotential player1 = model.createPotential();
player1.setDomain(d1);
player1.addFocalElement(fe11);
player1.addFocalElement(fe12);

//second potential ********************
```

```
//instantiate p1 domain
TBMVarDomain d2 = model.createDomain();
d2.addVariable(murderer);
d2.addVariable(weapon);
//create first focal element
TBMFocalElement fe21 = model.createFocalElement();
fe21.setDomain(d2);
fe21.addConfiguration(ColMustard, Candlestick);
fe21.setMass(0.8);
//create second focal element
TBMFocalElement fe22 = model.createFocalElement();
fe22.setDomain(d2);
fe22.addAllConfigurations();
fe22.setMass(0.2);
//create potential with masses
TBMPotential player2 = model.createPotential();
player2.setDomain(d2);
player2.addFocalElement(fe21);
player2.addFocalElement(fe22);
```

The knowledge of the first player will create the Potential graph presented in Figure 1.1 in the ontology's ABox:

This graph is a simplification, as there are some other relationships not depicted here. Some of the simplifications for space and clarity were: the types of some of the nodes, the domain node of the Potential and the Focal Elements, and the representation of the frame of discernment (Ω), which in reality should be several configurations, one for each possible combination of the instances of the variables in the domain. Player two's potential has a similar graph but the domain is different.

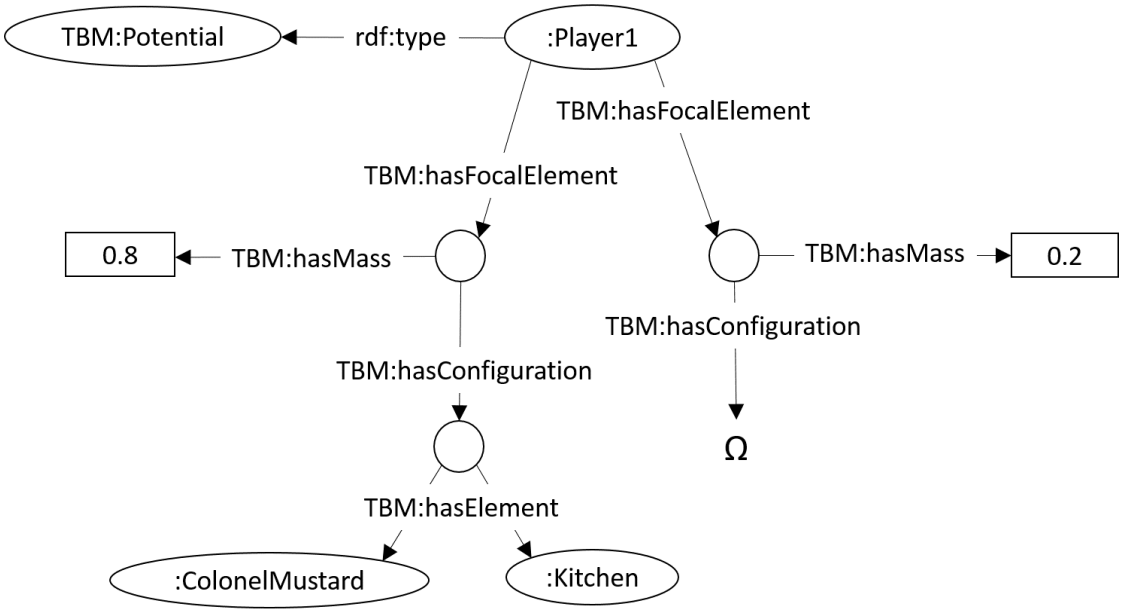

Fig. 1.1: TBM graph of the knowledge from player one

A.4 Combination and Interrogation

Once we have the two players' knowledge set up, we can combine the two Potentials and get the combined knowledge in a new Potential. The result potential can then be interrogated. In this case we want to know the belief that {`:ColonelMustard, :Kitchen, :Candlestick`} is the solution. Please note that the global potential will have a Domain of {`:Murderer, :Room, :Weapon`}

```
//Combine the two Focal Elements
TBMPotential finalPotential = model.combine(ont + "#finalPotential", player1,
    player2);

//instantiate global domain
TBMVarDomain d = model.createDomain();
d.addVariable(murderer);
d.addVariable(room);
d.addVariable(weapon);
```

```
//create the query focal element
TBMFocalElement query = model.createFocalElement(ont + "#queryFocalElement");
query.setDomain(d);
query.addConfiguration(ColMustard, Kitchen, Candlestick);

//Commit changes to the DB
if (model.supportsTransactions()) {
     model.commit();
}

//Query the system and print
System.out.println("Belief: " + finalPotential.bel(query));
System.out.println("Plausibility: " + finalPotential.pls(query));
System.out.println("Doubt: " + finalPotential.dou(query));
System.out.println("Ignorance: " + finalPotential.ign(query));
```

This will produce the following output on the computer's Standard Output:

```
Belief: 0.64
Plausibility: 1.0
Doubt: 0.0
Ignorance: 0.36
```

With the SPARQL hooks, it is also possible to interrogate the knowledge using SPARQL queries. This also opens the possibilities of using the belief functions with SPIN rules [17] for more advanced reasoning with incomplete knowledge.

```
String queryString
          = "PREFIX TBM: <http://mmv.eecs.qmul.ac.uk/TBM.owl#>\n"
          + "PREFIX CLUE: <http://mmv.eecs.qmul.ac.uk/CLUE.owl#>\n"
          + "SELECT ?p ?fe ?bel ?pls ?dou ?ign\n"
          + "WHERE {\n"
```

```
            + "   bind(CLUE:finalPotential as ?p) .\n"
            + "   bind(CLUE:queryFocalElement as ?fe) .\n"
            + "   bind(TBM:bel(?p, ?fe) as ?bel) .\n"
            + "   bind(TBM:pls(?p, ?fe) as ?pls) .\n"
            + "   bind(TBM:dou(?p, ?fe) as ?dou) .\n"
            + "   bind(TBM:ign(?p, ?fe) as ?ign) .\n"
            + "}";

// Execute the query and obtain results
Query q = QueryFactory.create(queryString);
try (QueryExecution qe = QueryExecutionFactory.create(q, model)) {
      ResultSet results = qe.execSelect();
      // Output query results
      ResultSetFormatter.out(System.out, results, q);
}
```

The output of this would be:

p	fe	bel	pls	dou	ign
CLUE:finalPotential	CLUE:queryFocalElement	0.64000000000000001e0	1.0000000000000002e0	-2.2204460492503131E-16	0.36000000000000001e0

Please note that the SPARQL formatter is introducing some very small rounding errors.

Appendix B

Riot Detection Dataset

The videos that comprise the dataset where acquired by MMV at QMUL thanks to it's involvement in the LASIE Project [66] of which the London MET police is also a partner. Given that two of the use cases in the project involve riots (logo and behaviour detection), the MET has kindly shared the data in a legal, ethical, and privacy compliant information sharing agreement framework.

In total, more than 3 terabytes of videos where shared by the MET. But for this work, 149 videos where selected. They are a total of 14,6 gigabytes, and are composed by 1,562,516 frames.

All the videos where given a 32 chars long hexadecimal name. The videos are encoded with a variety of codecs, but most of them are mpeg4/AVC videos. The also come in a variety of resolutions, but most of them are 704x288 or 704x576. Most of the videos are 12 frames per second. The average length of the videos is 14.6 seconds, but there are some videos that are an hour long, while a few others are only a few frames.

The ground truth annotation consists of XML files generated by the Viper GT tool [65]. The annotations were performed on the frames extracted from the videos. The annotation consists on the presence of 6 concepts related to riots. The concepts have

Fig. 2.1: Example of "Running"

all been defined in Chapter 5. Figures 2.1 to 2.7 present example frames for each of the concepts. All faces in the frames have been pixelated to protect the identity of those involved.

Viper GT uses it's own XML format for the annotation. More information on the schema employed by Viper GT can be seen in [65]. The Viper GT Schema used for this task consisted of an OBJECT descriptor called "Concepts" comprised of 6 different dynamic attributes, one for each relevant concept. Only one "Concept" record is needed for each frame. This concept record indicates the presence or absence of the different concepts.

We are making publicly available this ground truth data at `http://www.eecs.qmul.ac.uk/~chps3/riotDetectionDataset.zip`. No personal identifiable information is present in the annotations to protect the identity of the people in the videos.

Fig. 2.2: Example of “Face Covered”

Fig. 2.3: Example of “Crowd”

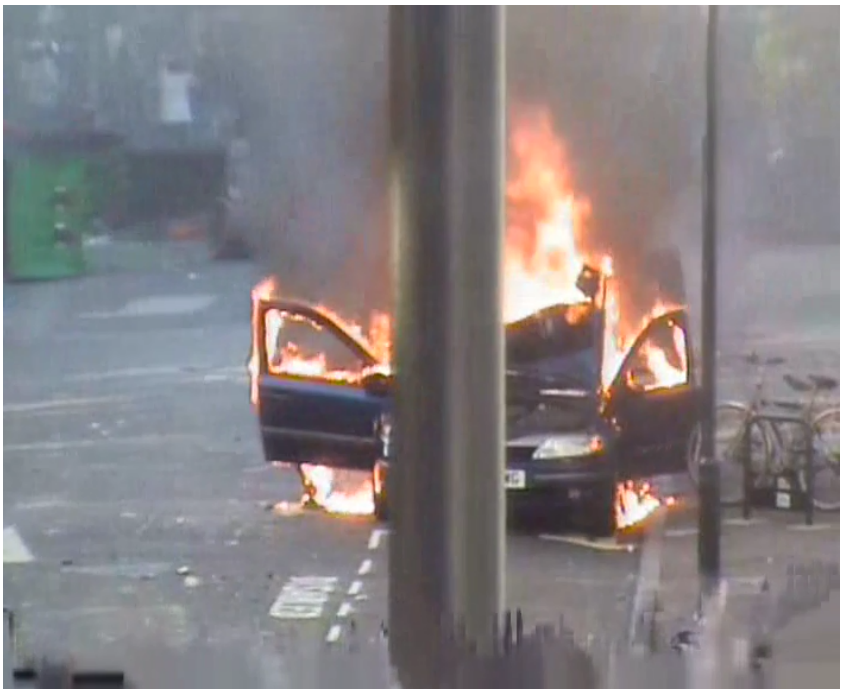

Fig. 2.4: Example of “Fire”

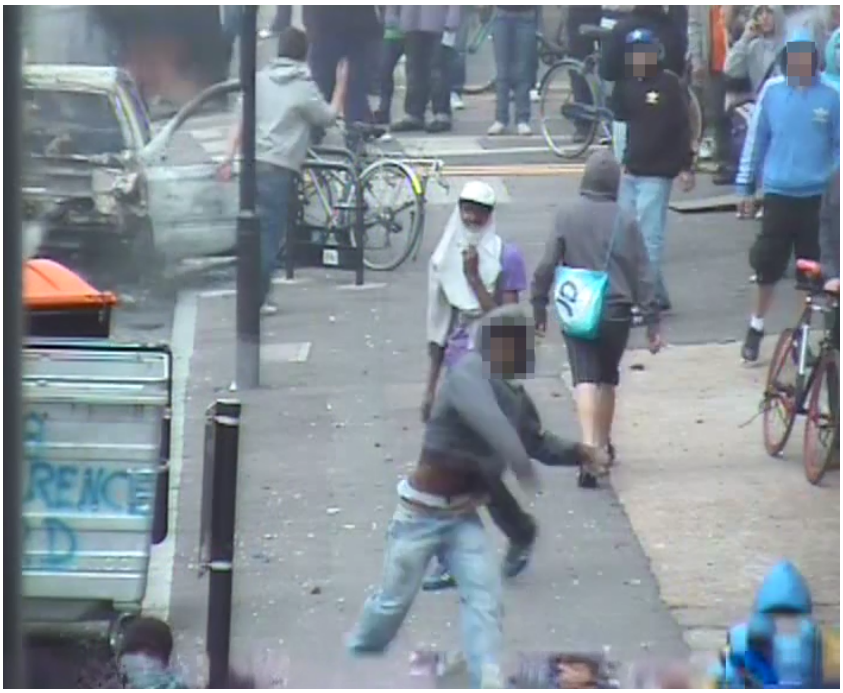

Fig. 2.5: Example of “Vandalism”

Fig. 2.6: Example of “Riot”

Fig. 2.7: Example of a frame without an event