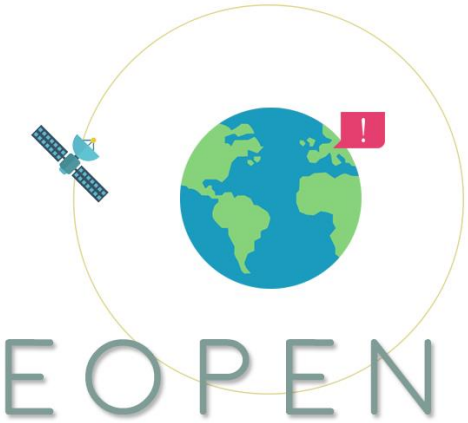

EOPEN

opEn interOperable Platform for unified access and analysis of Earth observatioN data

H2020-776019

D5.1

The EOPEN ontology and semantic reasoning support

Dissemination level:	Public
Contractual date of delivery:	Month 19, 31 May 2019
Actual date of delivery:	Month 19, 28 May 2019
Workpackage:	WP5 Semantic representation and the EOPEN ontology
Task:	T5.1 The EOPEN ontology, T5.2 Linked open EO data, T5.3 Reasoning for decision support
Type:	Other
Approval Status:	Approved
Version:	1.0
Number of pages:	55

Filename:	d5.1-The EOPEN ontology and semantic reasoning support_v1.0.docx

Abstract

This deliverable documents the semantic models for mapping the EOPEN-pertinent conceptualisations on ontology-related constructs. In addition, it describes the functionality of the first version of semantic integration and reasoning techniques. First, the purpose, scope, intended users and the requirements of the ontologies as identified at this phase of the project are described. Their specification has been driven by the WP2 initial user requirements identified for the individual scenarios, as well as by the dependencies incurring from the interaction with the WP3 and WP4 analysis components and the WP6 functionality aspects. Second, the literature is reviewed, covering state-of-the-art languages for formal knowledge representation, existing ontologies covering domains and requirements relevant to those of EOPEN, and methods pertinent to the extraction of location and organisation entities encountered in user tweets. Third, the current status of the EOPEN ontologies and tools is described, discussing the main entities they comprise. Fourth, the basic principles that underpin the first preliminary version of the WP5 reasoning framework towards reasoning and interpretation are described. The report also describes the progress on developing the Linked Data infrastructure to connect EO and non-EO data. Last, the report presents examples of the created annotation models.

co-funded by the European Union

History

Version	Date	Reason	Revised by	Approved by
0.1	03/04/2019	ToC	Georgios Meditskos, Stefanos Vrochidis, Ioannis Kompatsiaris	
0.2	02/05/2019	Contribution to SoA and methodology sections	Georgios Meditskos	
0.3	15/05/2019	Contribution to Ontology sections	Georgios Meditskos	
0.4	15/05/2019	Contribution to localisation section	Ilias Gialampoukidis	
0.5	16/05/2019	Contribution to EO data section	Bernard Valentin, Leslie Gale	
0.6	17/05/2019	Integration of different sections	Stefanos Vrochidis	
0.7	20/05/2019	Final version for internal review	Georgios Meditskos	
0.8	22/05/2019	Internal review	Vasileios Sitokonstantinou	
1.0	24/05/2019	Final version	Ioannis Kompatsiaris	
	29/05/2019			G. Vingione

Author list

Organisation	Name	Contact Information
SPACEAPPS	Bernard Valentin	bernard.valentin@spaceapplications.com
SPACEAPPS	Leslie Gale	leslie.gale@spaceapplications.com
CERTH	Georgios Meditskos	gmedisk@iti.gr
CERTH	Ilias Gialampoukidis	heliasgj@iti.gr
CERTH	Stefanos Vrochidis	stefanos@iti.gr
CERTH	Ioannis Kompatsiaris	ikom@iti.gr

Executive Summary

The present deliverable reports on the work carried out within T5.1, T5.2 and T5.3, relevant to the development of the EOPEN ontologies and the representation and mapping of content on ontological entities (T5.1). In addition, it describes the first preliminary framework towards reasoning (T5.3) and elaborates on the roadmap for linking of open EO and non-EO data (T5.2).

More specifically, the present deliverable presents the current content of the EOPEN ontologies and the methodology adopted to build them. Based on the requirements set forth by WP2 and the dependencies incurring from the interaction with the other WPs, the purpose, scope, intended users and uses, and the requirements of the EOPEN ontologies were identified. These specifications, along with the modelling insights from the relevant literature, served as guidelines for building the first version of the EOPEN ontologies that currently comprises modules for capturing the analysis results (metadata) of social media analysis (non-EO data), such as topics and localisation.

In addition, we present a preliminary version of the reasoning layer whose purpose is to enrich the supported semantics and metadata both at the terminological level, by defining additional class and property axioms, and at the assertional level by incorporating inference rules. The additional inference capabilities will afford the contextual enrichment of the knowledge graphs that will be useful at query-time, i.e. for context-aware tweet retrieval. Finally, a roadmap is presented on how EO and non-EO will be interlinked in the knowledge base in order to facilitate advanced context awareness and support the modelling and reasoning requirements set in the project.

The work presented within this document presents the preliminary version of the EOPEN ontologies, reasoning and interpretation framework. More elaborate ontology-based interpretation and reasoning tasks will be tackled in future versions of the framework and reported in upcoming deliverables.

Abbreviations and Acronyms

CQ	Competency Questions
CRF	Conditional Random Fields
DL	Description Logic
DLP	Description Logic Programs
HMM	Hidden Markov Models
KB	Knowledge Base
LSTM	Long Short-Term Memory
NER	named entity recognition
OGC	Open Geospatial Consortium
ORSD	Ontology Requirements Specification Document
OTK	On-To-Knowledge
OWL	Web Ontology Language
RDF	Resource Description Framework
RML	RDF Mapping language
RNN	recurrent neural networks
SPIN	SPARQL Inferencing Notation
SVM	Support Vector Machine
SWRL	Semantic Web Rule Language
SWRL	Semantic Web Rule Language
UPON	United Process for Ontologies
WGS	World Geodetic System

Table of Contents

1 INTRODUCTION

One of the cardinal objectives of WP5 is to provide the framework for encoding, aggregating (T5.1), semantically interlinking (T5.2) and analysing information (T5.2) relevant to the EOPEN application domain. In particular, WP5 provides the knowledge structures and vocabularies (ontologies) for defining a flexible and modular ontology-based framework for representing (a) Earth Observation, (b) meteorological and climate information, (c) social media information, (d) domain-specific aspects (e.g. floods, food security). The ontological framework will consist of a network of interconnected ontologies, along with a set of appropriate tools for populating the ontologies. The models created will constitute the foundations for the reasoning mechanisms. The reasoning framework will take into account underlying context coming from other EOPEN modules as well as domain knowledge.

The logical dependencies of WP5 with the other WPs of the EOPEN project are depicted in Figure 1. The figure also depicts the dependencies with WP2, WP6 and WP7 relevant to the development of the modules that will be integrated in the system and the feedback needed from the users with respect to requirements and evaluation.

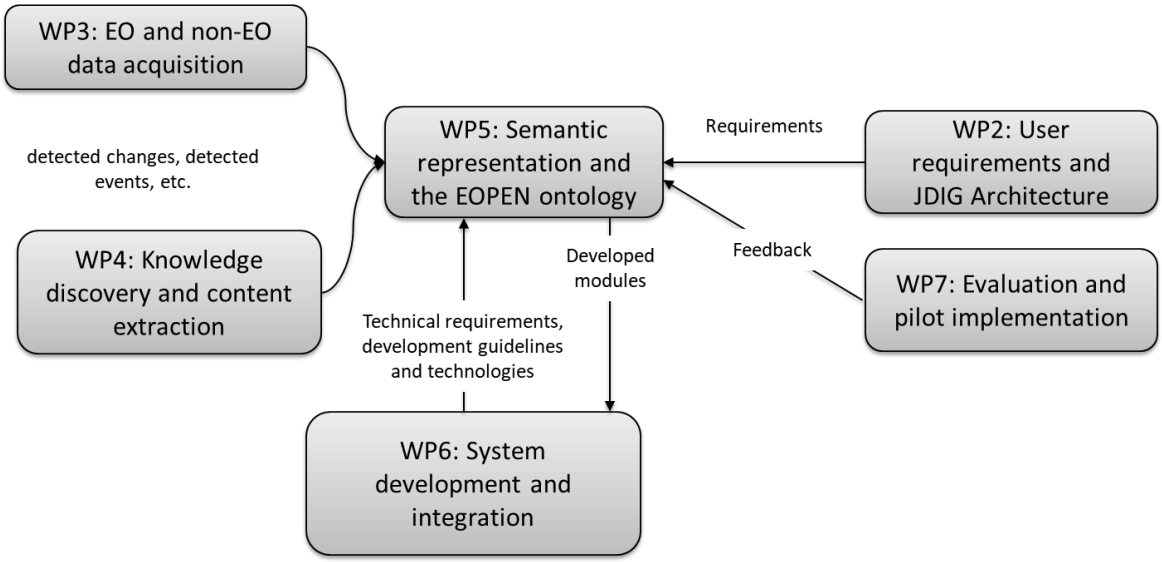

Figure 1 Logical dependencies of WP5 with the other WPs

In order to promote interoperability, extensibility and sharing, WP5 reuses and extends existing standards for defining the vocabulary of the annotations, as well as the patterns for associating these vocabularies with the generated assets. More specifically, the metadata vocabularies are defined in the Web Ontology Language (OWL 2[1] [12]), the W3C standard for defining and sharing ontologies. Similarly, the metadata are associated with assets using the Web Annotation Data Model [47], which provides an extensible and interoperable framework for expressing annotations. This model was published by the W3C Web Annotation Working Group as a Recommendation (since 23 February 2017), describing a common approach to express annotations in a manner that is simple and convenient, while at the same time enables more complex requirements. For geospatial data, WP5 capitalises on GeoSPARQL [39] an extension of SPARQL which is used to offer representation and

[1] https://www.w3.org/TR/owl-profiles/

querying in geospatial data. This standard was proposed by Open Geospatial Consortium (OGC).

The population of the EOPEN ontologies is done automatically by mapping the information provided as input by other components of the system. To this end, WP5 develops the necessary algorithms and interfaces for the structural and semantic mapping of data among different schemas and vocabularies, creating interlinked RDF-based knowledge structures pertinent to the assets derived by the EOPEN modules. For geospatial data, we use GeoTriples [24], otherwise the mappings are performed procedurally by custom-developed services.

Finally, WP5 provides the reasoning layer, whose purpose is to address WP5's reasoning requirements, in the form of ontology-based axiomatisations (e.g. complex class descriptions and property axioms), inference rules and data-driven analytics (e.g. for tweet localisation). The underlying reasoning techniques will afford the derivation of data- and knowledge-driven interpretations, enabling the system to abstract from incoming information and enrich the underlying knowledge graphs. This combination of semantically rich and interlinked knowledge graphs will foster the retrieval of data based on semantic relationships and not simply on keyword-based search.

Figure 2 presents the conceptual architecture of WP5 that consists of the following entities:

- Knowledge Base (KB), that provides native RDF storage and querying services
- Population, which implements the mapping services of input provided by other components, implementing linked data design principles
- Reasoning and context enrichment, which semantically enriches the content of the KB
- Localisation for extracting locations mentioned in tweets
- Linked Data, which implements linked data design principles to further enrich the derived knowledge

The remainder of this document is structured as follows: Section 2 overviews the adopted methodology for the creation of the EOPEN ontologies. Section 3 describes the user requirements that are relevant to WP5 modelling and reasoning framework. Section 4 reports the modelling specifications of the EOPEN ontologies. Section 5 reviews the relevant literature for reasoning, ontologies, named entity recognition and annotation models. Section 6 presents the first version of the EOPEN ontologies guided by the specifications (section 4), along with the modelling insights derived from the literature analysis (section 5). Section 7 describes the basic principles that underpin the preliminary version of WP5's reasoning framework towards intelligent reasoning, context enrichment and localisation services. Section 8 presents how the created models are applied on a set of examples for validation purposes, while Section 9 discusses the results and concludes the document.

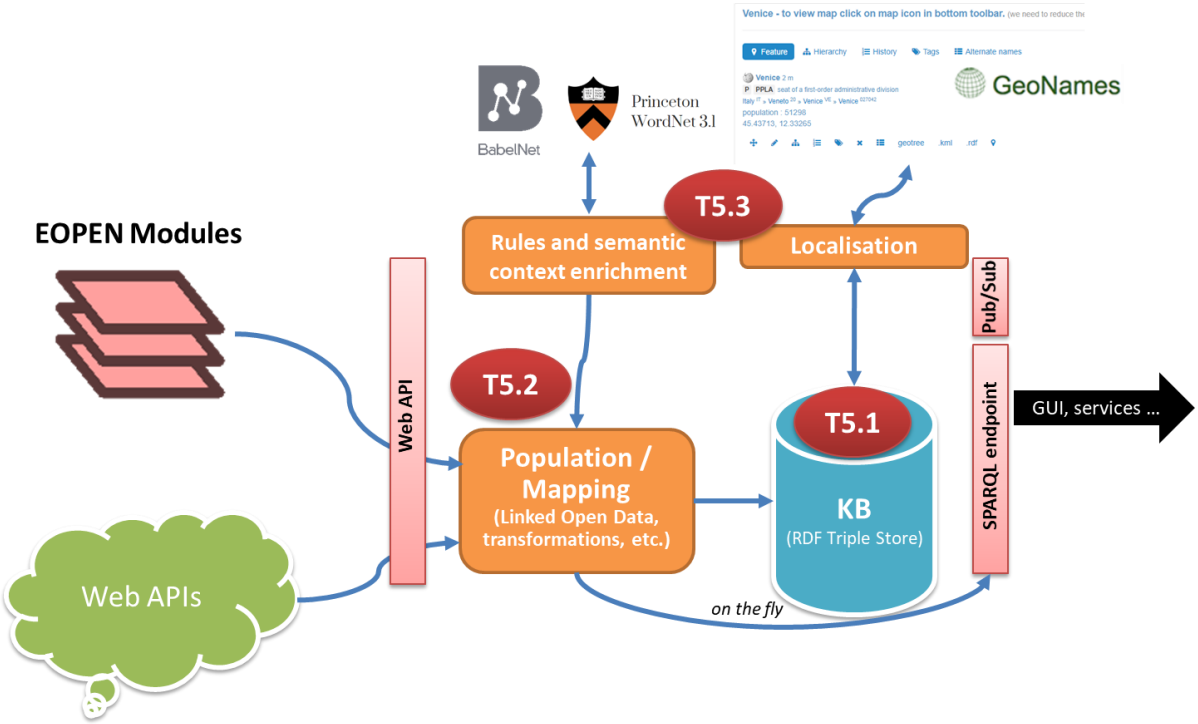

Figure 2 Conceptual architecture of the modules involved in WP5

2 METHODOLOGY FOR MODELLING REQUIREMENTS

2.1 Ontology development 101 methodology

There are many ways to model a domain using ontologies and the ontology development is essentially an iterative process. In this sense, there are several methodologies for ontological engineering such as On-To-Knowledge (OTK) [53], METHONTOLOGY [10], United Process for Ontologies (UPON) [34] and Ontology Development 101 [35]. Most of these methodologies introduce common features and development guidelines.

For the purposes of the EOPEN ontological framework, we adopted the methodology of *Ontology Development 101* which consists of the following iterative steps:

Step 1. Determination of the domain and scope of the ontology
Step 2. Reuse of existing ontologies
Step 3. Enumeration of important terms
Step 4. Definition of the classes and the class hierarchy
Step 5. Definition of the properties
Step 6. Creation of instances.

In literature, the determination of the domain and scope of the ontology can be documented in a template-based report called "Ontology Requirements Specification Document" (ORSD) [55]. This document allows the systematic specification of "why the ontology is being built", "what its intended uses are", "who the end-users are", and "which requirements the ontology should fulfil". In particular, the ORSD report contains the following fields:

1. **Purpose**: the main general goal of the ontology (i.e. how the ontology will be used in EOPEN)
2. **Scope**: the general coverage and the degree of detail of the ontology
3. **Implementation language**: the formal language of the ontology
4. **Intended end-users**: the intended end-users expected for the ontology
5. **Intended uses**: the intended uses expected for the ontology
6. **Ontology requirements**
    a. **Non-functional requirements**: the general requirements or aspects that the ontology should fulfil, including optional properties for each requirement
    b. **Functional requirements**: the content specific requirements that the ontology should fulfil in the form of groups of competency questions and their answers, including optional priorities for each group and for each competency questions
7. **Pre- Glossary of terms**
    a. **Terms from competency questions**: the list of items included in the competency questions and their frequencies
    b. **Terms from answers**: the list of terms included in the answers and their frequencies
    c. **Objects**: the list of objects included in the competency questions and their answers

Before presenting the EOPEN ORSD (section 4), we outline the WP5 relevant application context within which the EOPEN ontology is deployed (section 3).

3 USER REQUIREMENTS RELEVANT TO ONTOLOGIES AND REASONING

This section presents the application context relevant to WP5, describing relevant user requirements that drive the development of the EOPEN modelling and reasoning framework. To this end, we have investigated the description of the context and the users for each scenario, as well as the requirements that have been presented in D2.2 "User requirements". These requirements are translated into technical requirement in D6.1 "System Requirements and Architecture". Table 1 presents the user requirements that are relevant to the WP5 representation and reasoning framework, briefly describing the main functionalities and services that need to be supported.

Table 1 User requirements relevant to WP5 representation and reasoning framework

User Requirement ID (D2.2)	Description	WP5 Relevance / Dependency
PUC1_GA2	As an environmental manager I want to retrieve real-time data	• Support searching functionality over the KB to retrieve recently collected metadata
PUC1_GA3	As an environmental manager I want to retrieve semantic information	• Support searching functionality over the KB with the collected metadata
PUC1_GA7	As an environmental manager I want to retrieve the history of each event	• Support searching functionality and interface over the KB to get historical metadata which refer to an event
PUC2_GB4	As an administrator I want to access and manage satellite data	• Support searching functionality over the KB with the collected metadata • Provide the ability to manage the metadata over the KB
PUC3_GC2	As an administrator I want to easy access and manage the datasets	• Support searching functionality over the KB with the collected metadata • Provide the ability to manage the metadata over the KB
PUC3_GC5	As an administrator I want to select an area of interest	• Provide the ability to retrieve the collected metadata which refer to an area of interest from the KB
PUC3_GC7	As an administrator I want to browse historical operations	• Support searching functionality and interface over the KB to find the history of metadata
PUC3_GC12	As a user I want to retrieve relevant tweets with specific keywords	• Support searching functionality and interface over the KB to get metadata that contain specific keywords

4 ONTOLOGY REQUIREMENT SPECIFICATION DOCUMENT

This section presents the ORSD which provides the specification of the EOPEN ontological framework. The ORSD may be further elaborated and extended as the system functionalities will evolve to take into account hidden and unrevealed aspects that are not covered by the current ontology requirements

EOPEN ORSD	
1	**Purpose**
	The purpose of the EOPEN representation framework is to provide the ontological structures and vocabularies (ontologies) to capture the results of the EOPEN analysis modules in a reusable and interoperable manner. To this end, the ontological framework will provide the annotation model needed in order to support data modelling, integration and reasoning over the distilled information. These include: • Constructs for capturing metadata of non-EOP data, such as tweet-related information, event detection, etc. • A structured model and format to enable annotations and assertions to be defined, shared and reused across both within the EOPEN application context but also within different hardware and software platforms.
2	**Scope**
	The EOPEN ontology has to formally capture: • Social media information derived from twitter crawling and results analysing. • Localisation information derived from the location extracted from the tweet. • Topic information derived from the combination and analysis of the extracted information. • Events information derived from the event detection analysis. • Roads, rail and waterway information derived from satellite images. • Rice crop information derived from satellite images. A key design choice underpinning the engineering of the EOPEN ontologies is the adherence to a pattern-based approach, so as to capitalise on a modular, extensible and interoperable framework for expressing annotations and achieve a better degree of knowledge sharing, reuse and interoperability.
3	**Implementation Language**
	The ontology will be implemented in OWL 2 [12], the officially recommended language by W3C for knowledge representation in the Semantic Web.
4	**Intended End-Users**
	The EOPEN system considers different types of end users depending on the application context, who will interact with the generated knowledge through

	authoring tools: • ***PUC1: Administration, related to civil protection and environment*** **Administration offices and firefighters:** Administration offices related to civil protection or environmental issues and firefighters who want to be aware of real-time events and historical data of events to manage critical issues (eg. fire, flood). • ***PUC2: Decision making, related to food security in South Korea*** **Research Institutes and Science Centres:** Korea Rural Economic Institute and the National Institute of Agricultural Science who want to access food security related information such as rice maps and rice yield estimates. KREI and NIAS then share such relevant research findings and needs of producers and consumers to the government institutes which then deliver it to the Congress of South Korea. • ***PUC3: System repairs and reindeer herding environments*** **Government agencies and reindeer researchers:** Government agencies and reindeer researchers who want to access data to manage system repairs (e.g. road, rail and waterway) or select the ideal environment for reindeer herding.
5	**Ontology Requirements**
	Non-functional requirements
	NFR1. The ontology should adopt available standards whenever possible and reuse existing ontologies and vocabularies
	Functional requirements: Groups of competency questions
	The list of Competency Questions (CQ) below has been derived by studying the Pilot Use Case scenarios and user requirements. The questions have been also elicited through the direct interaction with technical partners. To this end, a simulation example has been carried out in order to collect additional technical and user-related requirements that drive the development of the annotation models. **Tweets** **CQ1** Which is the identifier of the tweet? **CQ2** Which is the tweet URL? **CQ3** Is the tweet a top ranked tweet? **CQ4** Which is the latitude of the tweet? **CQ5** Which is the longitude of the tweet? **CQ6** In which city does the tweet refer to? **CQ7** Which is the location of the user who wrote the tweet? **CQ8** Which is the dbpedia URL for the location of the user? **CQ9** Which is the dbpedia URL for the location where the tweet refers to? **CQ10** Which are the labels found in the tweet? **CQ11** Which is the accuracy of each label? **CQ12** Which is the name of each label?

	CQ13 Which is the annotation of a tweet? **CQ14** In which cluster does the label belong? **CQ15** What other labels belong in the same cluster? **CQ16** In which collection does the annotation belong to? **CQ17** What other annotations are part of this collection? **CQ18** Which is the resource that the annotation is connected with? **CQ19** Which is the cluster annotation of a collection? **CQ20** Which is the cluster annotation of a cluster body? **CQ21** Which is the time of the tweet? **CQ22** Which is the date of the tweet? **CQ23** Which is the topic of the tweet? **CQ24** Which is the event of the tweet? **CQ25** What are the hashtags of the tweet? **CQ26** What are the mentions/tags of the tweet? **CQ27** Is there an image in the tweet? **CQ28** Is there a video in the tweet? **CQ29** Is there a URL in the tweet? **CQ30** How many people reposted the tweet? **CQ31** How many people marked the tweet as favorite? **CQ32** How many people replied to the tweet? **CQ33** How many people liked the tweet? **CQ34** How many followers does the user (who posted the tweet) have? **CQ35** Which is the user's (who posted the tweet) profile URL? **CQ36** Which is the user's (who posted the tweet) username? **CQ37** Are there any emojis in the tweet? **Events** **CQ38** Which events were detected? **CQ39** Which is the geometry polygon the event refers to? **CQ40** Which is/are the point/-s the event refers to? **CQ41** Which words does the event contain? **Topics** **CQ42** Which topics were detected? **CQ43** Which is the geometry polygon the topic refers to? **CQ44** Which is/are the point/-s the topic refers to? **CQ45** Which words does the topic contain? **CQ46** Which is the BabelSynset identifier for each term? **CQ47** Which is the BabelNet URL for each term? **CQ48** Which is the Babelfy source for each term? **CQ49** Which is the dbpedia URL for each term? **CQ50** Which is the Babelfy score for each term? **CQ51** Which is the Babelfy coherence score for each term? **CQ52** Which is the Babelfy global score for each term? **CQ53** Which is the WordNet identifier for each label in topic detection? **CQ54** Which is the level 1 hypernym WordNet identifier for each term? **CQ55** Which is the level 2 hypernym WordNet identifier for each term?

	CQ56 Which is the level 3 hypernym WordNet identifier for each term? **CQ57** Is there another term with the same level 1 WordNet identifier? **CQ58** Is there another term with the same level 2 WordNet identifier? **CQ59** Is there another term with the same level 3 WordNet identifier? **Location** **CQ60** What is the weather like in a location? **CQ61** What is the temperature of a location? **CQ62** What is the climate of a location? **Fields and policies** **CQ63** Which is the identifier of the field? **CQ64** In which satellite image is the field depicted? **CQ65** Does the field contain rice? **CQ66** Does the field conform to the agricultural policies which are applicable in this area? **CQ67** Does the field conform to environmental rules? **CQ68** Which is the geolocation of a field? **System repairs management** **CQ69** Which is the identifier of the image? **CQ70** In which satellite image is the road depicted? **CQ71** Does the road need maintenance? **CQ72** Which is the geolocation of the road? **CQ73** In which satellite image is the rail depicted? **CQ74** Does the rail need maintenance? **CQ75** Which is the geolocation of the rail? **CQ76** In which satellite image is the waterway depicted? **CQ77** Does the waterway need maintenance? **CQ78** Which is the geolocation of the waterway? **CQ79** Is the image inside an area of interest? **CQ80** Which metadata refers to the area of interest? **CQ81** What is the history of an area of interest?

The competency questions cover a very exhaustive list of aspects relevant to the EOPEN domain. Though the initial set of EOPEN ontologies will not cover this extended list and depth of detail, they provide the modular structures that will enable the future extensibility of the model.

5 STATE OF THE ART

This section provides an overview on the relevant state of the art with respect to knowledge representation languages, already existing ontologies addressing project-relevant fields, as well as natural language approaches to address the localisation task. More specifically, we present the basics of Description Logic (DL) languages [2], on which the official W3C recommendation for creating and sharing ontologies in the Web (OWL 2) is grounded, the different OWL 2 species, as well as relevant rule-based languages. We then provide a briefly review on the representative ontologies that have been proposed in the literature for modelling core aspects relevant to the EOPEN application domain that fall into WP5's modelling requirements. Lastly, a brief presentation of the state-of-the-art in named entity recognition (NER) is included, which demonstrates the developments that drive the inner workings of the localisation procedure.

5.1 Web Ontology Language

In the literature, ontologies have been widely used as an effective way for modelling domain information because they can represent and organise information, context and relationships more accurately. In addition, they offer easy expandability by merging, expanding and combining parts of existing ontologies into new ones.

Ontologies are models used to capture knowledge about some domain of interest. Formally speaking, ontologies are *explicit formal specifications of shared conceptualizations* [16, 54]. They represent abstract views of the world including the objects, concepts, and other entities that are assumed to exist in some area of interest, their properties and the relationships that hold among them. Their expressivity and level of formalisation depend on the knowledge representation language used.

Within the Semantic Web, which is an extension of the current Web that aims to establish a common framework for sharing and reusing data across heterogeneous sources, ontologies play a key role. The Semantic Web vision is to make the semantics of web resources explicit by attaching to them metadata that describe meaning in a formal, machine-understandable way. In this effort, the Web Ontology Language [6] has emerged as the official W3C recommendation for creating and sharing ontologies on the Web. In the rest of this section, we present the basics of Description Logic languages, on which OWL semantics are grounded, the different OWL species, as well as relevant rule-based languages.

5.1.1 DL Reasoning

Description Logics [2] are a family of knowledge representation formalisms characterised by logically grounded semantics and well-defined reasoning services. The main building blocks are *concepts* representing sets of objects (e.g. `Field`), *roles* representing relationships between objects (e.g. `contains`), and *individuals* representing specific objects (e.g., `RiceCrop`). Starting from *atomic* concepts, such as `Field`, arbitrary complex concepts can be described through a rich set of *constructors* that define the conditions on concept membership. For example, the concept $\exists$`hasNeighbour.Field` describes those objects that are related through the `hasNeighbour` role with an object from the concept `Field`; intuitively, this corresponds to all those fields that are neighbours with at least one field. A DL knowledge base K typically consists of a *TBox* T (terminological knowledge) and an *ABox*

A (assertional knowledge). The TBox contains axioms that capture the possible ways in which objects of a domain can be associated. For example, the TBox axiom `RiceCrop` $\sqsubseteq$ `Crop` asserts that all objects that belong to the concept `RiceCrop`, are members of the concept `Crop` too. The ABox contains axioms that describe the real world entities through concept and role assertions. For example, `Field(field_id)` and `isLocated(field_id,Korea)` express that `RiceCrop` is a crop and it is located in Korea. Table 2 summarises the set of terminological and assertional axioms.

Table 2 Terminological and assertional axioms

Name	Syntax	Semantics
Concept inclusion	$C \sqsubseteq D$	$C^{\mathrm{I}} \subseteq D^{\mathrm{I}}$
Concept equality	$C \equiv D$	$C^{\mathrm{I}} = D^{\mathrm{I}}$
Role Equality	$R \equiv S$	$R^{\mathrm{I}} = S^{\mathrm{I}}$
Role inclusion	$R \sqsubseteq S$	$R^{\mathrm{I}} \subseteq S^{\mathrm{I}}$
Concept assertion	$C(\alpha)$	$\alpha^{\mathrm{I}} \in C^{\mathrm{I}}$
Role assertion	$R(\alpha,b)$	$(\alpha^{\mathrm{I}}, b^{\mathrm{I}}) \in R^{\mathrm{I}}$

5.1.2 DL reasoning services

DLs come with a set of powerful reasoning services, for which efficient, sound and complete reasoning algorithms with well understood computational properties are available. Example state-of-the-art implementations include Pellet [50] Racer [17], Fact++ [56] and Hermit [11].

Assuming a DL knowledge base $K = (T, A)$, typical reasoning services include:

- **Subsumption:** A concept C is subsumed by D in T (written $T \models C \sqsubseteq D$), *iff* $C^{\mathrm{I}} \subseteq D^{\mathrm{I}}$ for all interpretations I.
- **Equivalence:** Two concepts C and D are equivalent in T (written $T \models C \equiv D$) *iff* $C^{\mathrm{I}} \subseteq D^{\mathrm{I}}$ and $D^{\mathrm{I}} \subseteq C^{\mathrm{I}}$ *for all interpretations* I.
- **Disjoint:** A concept C is disjoint to a concept D in T *iff* in every interpretation I it holds that $C^{\mathrm{I}} \neq \emptyset$.
- **Consistency:** The ABox A is consistent w.r.t. T *iff* if there is an interpretation that is a model of both A and T.
- **Instance checking:** The individual α is an instance of C (w.r.t. K) (written $K \models C(\alpha)$) *iff* $\alpha^{\mathrm{I}} \in C^{\mathrm{I}}$ holds for all interpretations I of K.
- **Realisation:** The realisation of an instance α w.r.t. to K includes finding the most specific concepts C for which $a^{\mathrm{I}} \in C^{\mathrm{I}}$ holds for all interpretations I of K.

Hence, through subsumption one can derive the implicit taxonomic relations among the concepts of a terminology. For example, given the axiom `SownField` $\equiv$ `Field` $\sqcap$ $\exists$`contains.Crop`, one can infer that `Field` subsumes `SownField`.

Satisfiability and consistency checking are useful to determine whether a knowledge base is meaningful at all. Satisfiability checking enables the identification of concepts for which it is impossible to have members under any interpretation (for example, an unsatisfiable concept, though trivial, is `SownField` $\sqcap$ $\neg$ `SownField`). Consistency checking enables the

identification whether the set of assertions comprising the knowledge base is admissible with respect to the terminological axioms. For example if `EmptyField` and `SownField` are asserted as disjoint concepts, then the presence of both `SownField(field_id)` and `EmptyField(field_id)` leads to inconsistency.

Instance checking denotes the task of finding whether a specific individual is an instance of a given concept. Realisation of an individual, a more generic form of instance checking, returns all (most specific) concepts from the knowledge base that a given individual is an instance of. Its dual is the retrieval problem that given a specific concept *C*, it returns all individuals that belong to this concept. This reasoning service is the central to realise the task of recognition of situation types.

Falling under the classical logics paradigm, reasoning in DLs adopts the open-world assumption. Intuitively, if a fact α holds only in a subset of the models of the knowledge base *KB*, then we can conclude neither $KB \models \alpha$ nor $KB \models \neg\alpha$. For example, if the only available knowledge regarding the owners of a field is the assertion `isOwnedBy(Alice,field_id)`, we cannot deduce based on it alone that no one else owns the field. In contrast, formalisms adhering to the closed-world assumption make the common-sense conjecture that all relevant information is explicitly known, so all unprovable facts should be assumed not to hold. In our example, this amounts to concluding that Alice is the sole owner of this field. Hence, closed-world reasoning can be intuitively understood as reasoning where from $KB \models \alpha$, one concludes $KB \models \neg\alpha$. Such kind reasoning should not be confused however with closed domain reasoning, which involves reasoning only over explicitly known individuals.

5.1.3 **OWL and OWL 2**

The OWL is a knowledge representation language widely used within the Semantic Web community for creating ontologies. The design of OWL and particularly the formalisation of the semantics and the choice of language constructors have been strongly influenced by DLs. OWL comes in three dialects of increasing expressive power: OWL Lite, OWL DL and OWL Full. OWF Full is the most expressive of the three: it neither imposes any constraints on the use of OWL constructs, nor lifts the distinction between instances (individuals), properties (roles) and classes (concepts). This high degree of expressiveness comes however at a price, namely the loss of decidability that makes the language difficult to implement. As a result, focus has been placed on the two decidable dialects, and particularly on OWL DL, which is the more expressive of the two.

Despite the rich primitives provided for expressing concepts, OWL DL has often proven insufficient to address the needs of practical applications. This limitation amounts to the DLs style model theory used to formalise its semantics, and particularly the *tree model property* [57] conditioning DLs decidability. As a consequence, OWL can model only domains where objects are connected in a tree-like manner. This constraint is quite restrictive for many real-world applications, including the ambient intelligence domain, which requires modelling general relational structures.

Responding to this limitation and to other drawbacks that have been identified concerning the use of OWL in different application contexts throughout the years, the W3C working group produced OWL 2 [12]. OWL 2 is a revised extension of OWL, now commonly referred to as OWL 1. It extends OWL 1 with qualified cardinality restrictions; hence one can assert

for example that a social activity is an activity that has more than one actor: $\texttt{SocialActivity} \equiv \texttt{Activity} \sqcap \geq 2\texttt{hasParticipant.Person}$.

Another prominent OWL 2 feature is the extended relational expressivity that is provided through the introduction of complex property inclusion axioms (property chains). To maintain decidability, a regularity restriction is imposed on such axioms that disallow the definition of properties in a cyclic way. Hence, one can assert the inclusion axiom $\texttt{locatedIn} \circ \texttt{containedIn} \sqsubseteq \texttt{locatedIn}$ making it possible to infer that if a person is located for example in the bedroom of her house, then she is located in her house as well; however, it is not allowed to use both the aforementioned axiom and the axiom $\texttt{containedIn} \circ \texttt{locatedIn} \sqsubseteq \texttt{containedIn}$ as this leads to a cyclic dependency. Three profiles, namely OWL 2 EL, OWL 2 QL and OWL 2 RL, trade portions of expressive power for efficiency of reasoning targeting different application scenarios.

5.1.4 **Rules**

To achieve decidability, DLs, and hence OWL, trade some expressiveness for efficiency of reasoning. The tree-model property is one such example. It conditions the tree-shape structure of models, ensuring decidability, but at the same time it severely restricts the way variables and quantifiers can be used, dictating that a quantified variable must occur in a property predicate along with the free variable. As a result, it is not possible to describe classes whose instances are related to an anonymous individual through different property paths. To leverage OWL's limited relational expressivity and to overcome modelling shortcomings that OWL alone would be insufficient to address, a significant body of research has been devoted to the integration of OWL with rules.

A proposal towards this direction is the Semantic Web Rule Language (SWRL) [20], in which rules are interpreted under the classical first order logic semantics. Allowing concept and role predicates to occur in the head and the body of a rule without any restrictions, SWRL maximises the interaction between the OWL and rule components, but at the same time renders the combination undecidable. To regain decidability, several proposals have explored syntactic restrictions on rules [31, 46] as well as their expressive intersection of Description Logic Programs (DLP) [15]. The DL-safe rules introduced for example in [31] impose that rule semantics apply only over known individuals. It is worth noting that in practice DL reasoners providing support for SWRL actually implement a subset of SWRL based on this notion of DL-safety.

Taking a different perspective, a number of approaches have investigated the combination of ontologies and rules based on mappings of a subset of the ontology semantics on rule engines. For instance, [21] defines the *pD** semantics as a weakened variant of OWL Full, e.g., classes can be also instances, and they are extended to apply to a larger subset of the OWL vocabulary, using 23 entailments and 2 inconsistency rules. Inspired by the *pD** entailments and DLP, the semantics of the OWL 2 RL profile is realised as a partial axiomatisation of the OWL 2 semantics in the form of first-order implications, known as OWL 2 RL/RDF rules. User-defined rules on top of the ontology allow expressing richer semantic relations that lie beyond OWL's expressive capabilities, and couple ontological and rule knowledge.

SPARQL [18] is a declarative language recommended by the W3C for extracting and updating information in RDF graphs. It is an expressive language that allows the description of quite complex relations among entities. The semantics and complexity of the SPARQL query language have been fairly studied theoretically, showing that SPARQL algebra has the same expressive power as relational algebra [38] [19]. Although SPARQL is mostly known as a query language for RDF, by using the `CONSTRUCT` graph pattern, it is able to define SPARQL rules that can create new RDF data, combining existing RDF graphs into larger ones. Such rules are defined in the interpretation layer in terms of a `CONSTRUCT` and a `WHERE` clause: the former defines the graph patterns, i.e. the set of triple patterns that should be added to the underlying RDF graph upon the successful pattern matching of the graphs in the `WHERE` clause. The SPARQL Inferencing Notation (SPIN) [52] constitutes an effort to ease the definition and execution of SPARQL rules on top of RDF graphs. In SPIN, SPARQL queries can be stored as RDF triples together with any RDF domain model, enabling the linkage of RDF resources with the associated SPARQL queries, as well as sharing and reuse of SPARQL queries. SPIN supports the definition of SPARQL inference rules that can be used to derive new RDF statements from existing ones through iterative rule application.

5.2 Ontologies relevant to the EOPEN domain

In this section we describe the languages and extensions which are used for representing and querying geospatial data. We also describe some languages which transform data into RDF format and some tools for handling geospatial information.

5.2.1 Geospatial data representing, querying and saving

Since geospatial data are available in many cases via the Web in linked open data cloud [24], there comes the need to exploit them for decision making issues. This need emerged lately, as in the past the data that were available did not contain any geospatial information. The following languages are responsible for geospatial data representation and querying.

stRDF[23][24] is a new version of RDF for representing geospatial data that change over time. It is based in the philosophy of constraint databases and more specifically CSQL. stRDF means spatial/temporal RDF and is responsible for adding new datatypes to spatial and temporal literals by extending RDF which represents only thematic metadata. The development of stRDF contains two steps: development of sRDF and stRDF. sRDF is an RDF extension which allows the representation of spatial data except from thematic. stRDF is an RDF extension which allows the representation of thematic and spatial data with a temporal dimension.

Table 3 An example of using stRDF in GeoNames data that represent information about the greek town Olympia. Information contain data about geometry and burnt areas.

```
geonames:26 rdf:type dbpedia:Town.
geonames:26 geonames:name "Olympia".
geonames:26 strdf:hasGeometry "POLYGON((21 18,23 18,23 21,21 21,21 18));
        <http://www.opengis.net/def/crs/EPSG/0/4326>"^^strdf:WKT.
noa:BA1 rdf:type noa:BurntArea;
```

```
        strdf:hasGeometry "POLYGON((0 0,0 2,2 2,2 0,0 0))"^^strdf:WKT.
noa:BA2 rdf:type noa:BurntArea;
        strdf:hasGeometry "POLYGON((3 8,4 9,3 9,3 8))"^^strdf:WKT.
```

stSPARQL[23][24] is a new version of SPARQL for representing geospatial data. It adds spatial terms in SELECT, FILTER and HAVING and also supports update operations (INSERT, DELETE and UPDATE) of stRDF triples. Spatial terms may be spatial literals, query variables connected to spatial literals, results of set or geometric operations on spatial terms. stSPARQL is the querying language which is associated with stRDF language.

Table 4 An example for returning the names of towns that have been affected by fires.

```
SELECT ?name
WHERE {
?t a dbpedia:Town;
        geonames:name ?name;
        strdf:hasGeometry ?tGeo.
?ba a noa:BurntArea;
        strdf:hasGeometry ?baGeo.
FILTER(strdf:intersects(?tGeo,?baGeo))}
```

GeoSPARQL[24] is an extension of SPARQL which is used to offer representation and querying in geospatial data. This standard was proposed by *Open Geospatial Consortium* (*OGC*). It provides a vocabulary which can be used in geospatial RDF graphs and SPARQL queries. The vocabulary contains three main classes: ogc:SpatialObject which represents all feature and geometry objects that can be represented spatially, ogc:Feature which represents all feature objects and ogc:Geometry which represents all geometry objects. GeoSPARQL also offers an amount of properties for geometries handling in order to investigate the topological relationships among them. For example, ogc:equals, ogc:intersects, ogc:crosses, ogc:touches which return a Boolean value which corresponds to the geometries equality, intersection etc, while others eg. ogcf:distance, ogcf:intersection, ogcf:union return other value types (String, integer, double etc.) which correspond to the distance, intersection points etc.

5.2.2 Data transformation to RDF

Another factor that needs attention is that a lot of the available data are in formats other than RDF (eg. HTML). In order to exploit these data there comes the need to transform the non-RDF data which are available on the web into RDF format. Many frameworks have been developed lately for this scope and some of them are presented in this section. [4]

R2RML[2][44] is a W3C recommendation language which creates mappings by using existing relational data and converting to RDF data model. The structure and the vocabulary are

[2] https://www.w3.org/TR/r2rml/

defined by the user. The input is a relational database which conforms to a specific database schema. R2RML mappings are written in RDF Turtle syntax. R2RML mapping uses the logical tables (base table, view or valid SQL query) to retrieve data from the relational database. Triples map is a rule responsible for mapping the relational table to RDF by converting each row of the logical table into triples. Triples map rule contains a subject map, which creates all subjects IRIs from the logical table, and multiple predicate-object maps which consist of predicate and object maps. The result of this procedure are triples which are exported by the combination of subject, predicate and object maps.

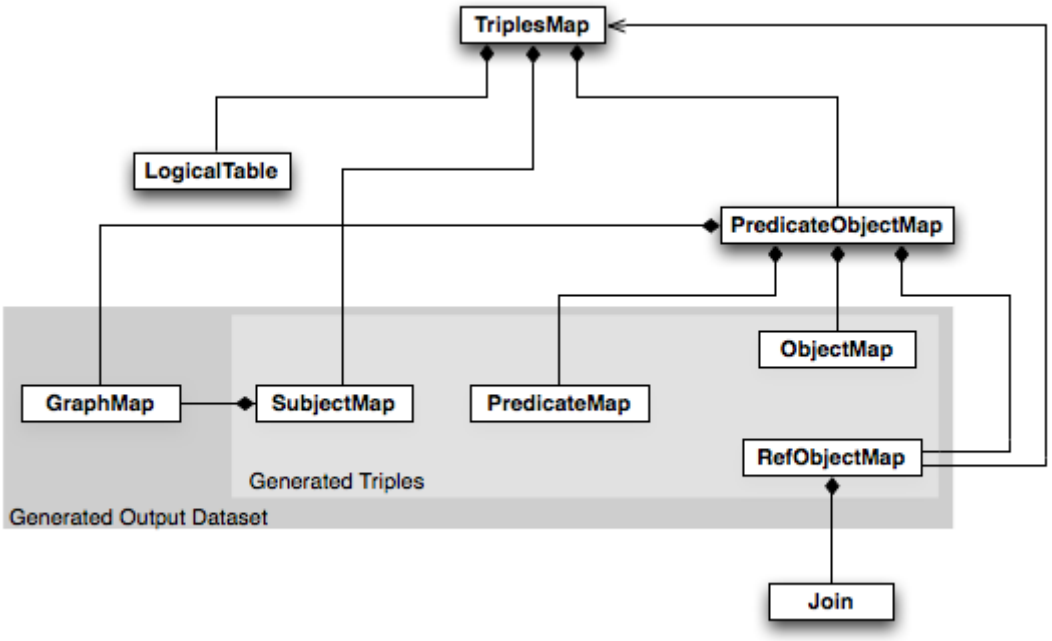

Figure 3 R2RML mapping overview

RDF Mapping language (RML)[3] [9][8] is a language responsible for converting data from multiple non-RDF formats into RDF. RML gets as input formats like XML, JSON, CSV, relational databases or RDF triplestores using SPARQL and creates a semantic representation of them by creating the appropriate rules or mappings. It provides a vocabulary for data defining while the user is able to define the iterator pattern about the way that the source data should be accessed.

[3] http://rml.io/

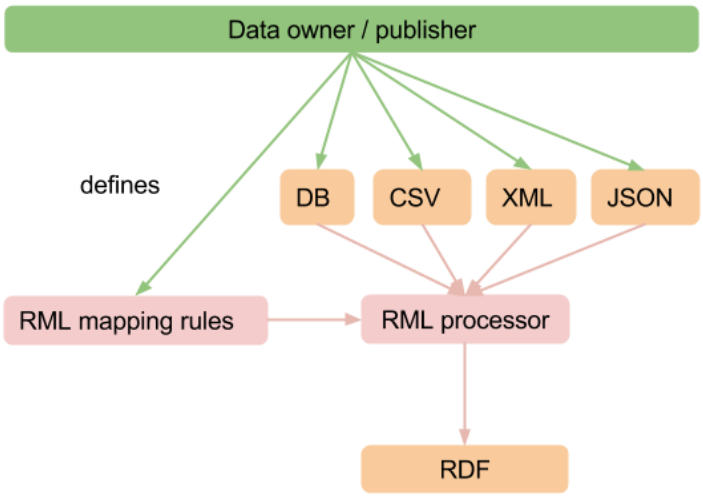

Figure 4 RML mapping overview

5.2.3 **Geospatial tools**

In this section we present some systems which are responsible for handling (converting, saving, querying) geospatial data. The tools provide a more complete solution, with enhanced functionality compared to the languages and standards that were presented in previous sections.

Geosensor [42] is a web-based system which detects change detection from satellite images using event detection from news items and social media content. Its architecture is based in semantic web technologies and consists of 11 components divided in three layers: change detection, event detection and semantic layer. Change detection layer is responsible for images downloading (Image Aggregator), storing (Hadoop Distributed File System) and change detecting (Change Detector) by computing pixels ratio from two images using the change detection algorithm. Event detection layer combines each land cover with an event and a possible explanation of that. In order to achieve that it scans specific news agencies and social media to collect news (News Crawler), stores news data (Apache Cassandra), processes stored results from Cassandra every half hour (Event Detector), combines news with the geolocation of the events (Lookup Service) and combines named entities which are extracted from events with a Flickr image, building a semantic web (Entity Extractor). Semantic layer is responsible for the connection between change detection layer and event detection layer. It uses multiple technologies: GeoTriples to convert geospatial data to RDF, Strabon to store them, SemaGrow for the SPARQL queries processing and Sextant as a web interface for handling (exploring, interacting, visualising) linked geospatial data.

GeoTriples[24] is a tool for transforming geospatial data from various formats into RDF. It comprises of three components: mapping generator, mapping processor and stSPARQL/GeoSPARQL evaluator. The mapping generator gets as input a file (ESRI Shapefiles, XML, GML, KML, JSON, GeoJSON and CSV documents or spatially-enabled relational databases (e.g., PostGIS and MonetDB)) and creates a mapping using RML and R2RML (rules) which is based on the GeoSPARQL vocabulary. As an optional step, the user has the opportunity to change this mapping according to his requirements. The mapping processor is the component which is responsible for the RDF graph generation by using the result of the mapping generator. The output can be expressed in a variety of RDF syntaxes such as Turtle,

RDF/XML, Notation3 or N-Triples. The stSPARQL/GeoSPARQL evaluator is a component for querying in a relational database using a R2RML mapping.

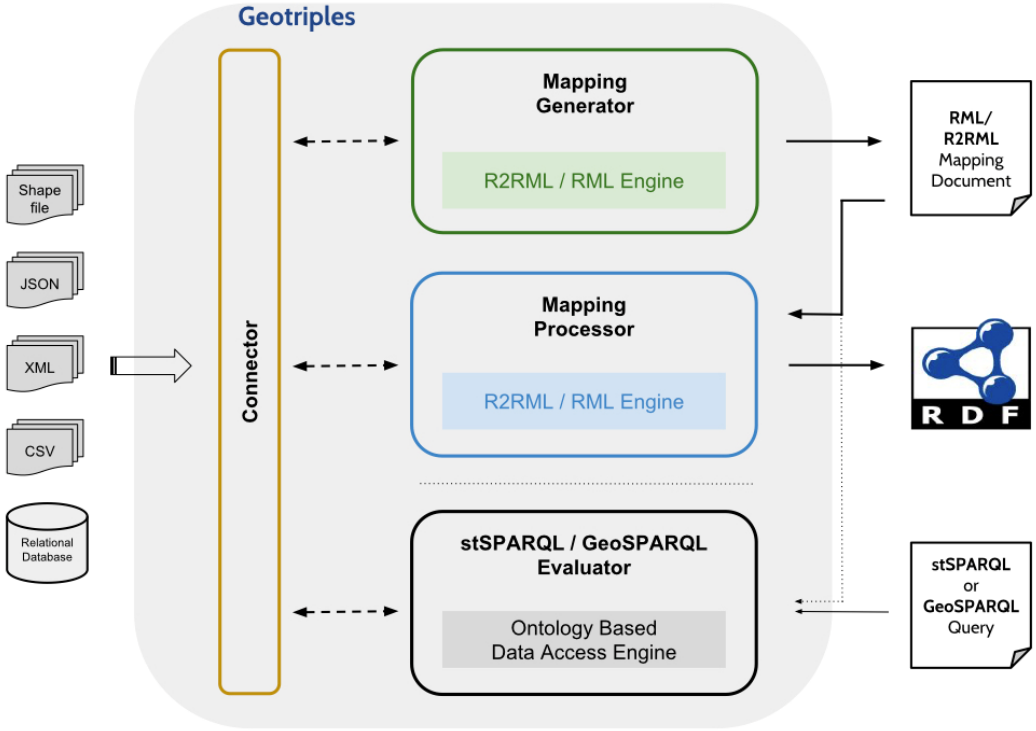

Figure 5 Geotriples tool architecture

Strabon[4][25] is an RDF triplestore for storing geospatial data that change over time. It supports two extensions of SPARQL: stSPARQL, which accesses data in stRDF form and GeoSPARQL, which queries geospatial data. It is mainly used to represent temporal dimensions among the data by using events or facts that change over time as it also offers some temporal functions. It extends Sesame RDF triplestore by supporting and handling thematic, spatial and temporal data. Strabon 3.0 version uses Sesame 2.6.3 version and has three modules: storage manager, query engine and POSTGIS. Storage manager creates two B+ tree two-column indices by receiving a predicate table. Query processing consists of four elements: a parser, an optimiser, an evaluator and a transaction manager. The parser generates an abstract syntax tree, which is mapped later and creates a query tree. Query tree is optimised according to Strabon optimisation techniques (stSPARQL extension functions and DBMS informing about spatial joins in stSPARQL queries) and sent to the evaluator to create the SQL query for evaluating PostgreSQL. The evaluator receives the results and carries out any needed operation. Results are formatted according to the selected data format.

Sparqlify[24] is a tool for converting SPARQL to SQL. It gives users the opportunity to search a relational database by querying data in SPARQL by using a mapping language which is similar to R2RML but has a different syntax. This opportunity removes the need of

[4] http://www.strabon.di.uoa.gr/

maintaining two datastores for both SPARQL and SQL queries execution, which is the main advantage of this method, but on the other hand it limits query flexibility in many cases. This drawback comes from the translation between SPARQL and SQL as limited SPARQL queries can be supported in SQL. Apart from SPARQL main features Sparqlify supports some geospatial features like st_intersects, which checks if two geometries have mutual points. Currently, Sparqlify supports access to the OpenStreetMap database as a linked data interface for the project LinkedGeoData.

Geometry2RDF[28] was a tool for transforming geospatial data into RDF. The tool took as input data stored in relational databases and produced as an output the RDF graph. The user could configure the properties that concerned him. Geometry2RDF tool was based in Jena and GeoTools libraries. Though the tool is no longer maintained by its developers, it was the first tool that was dealing with this issue and has been used as a basis to develop the first version of TripleGeo tool. Nowadays, Geometry2RDF[5] is offered as a library for generating RDF files. It is based in Jena and connects to Oracle geospatial databases.

TripleGeo[36] is a tool which can convert data from many type formats into RDF graph. Input data can be in relational database format (PostgreSQL/PostGIS, Oracle Spatial and Graph, MySQL and MS SQL Server) or raw files (ESRI shapefiles, GeoJSON, GML, KML, GPX and CSV). It contains three modes: graph, stream and RML mode. Graph mode is responsible for the transformation of the input data into RDF. Stream mode is handling each entry of the input data in a different way. RML mode exploits the RML mappings in order to convert the data. The disadvantage of this tool is that stream and graph modes contain only four attributes (ID, name, geometry and category) for each tuple. This fact results in losing an amount of information which is available in the dataset.

Ontop4theWeb[4] is a tool that supports querying data which are available on the Web in table or REST API format. The tool handles data as relational as it converts the web data to virtual table operators using SQL. It allows SQL querying by creating R2RML mapping rules to convert data in RDF format. When a SPARQL query is created, the query is transformed to SQL and results are returned in RDF format. A time window is available, where a user can retrieve data. If time passes, data need to be re-imported. Ontop4theWeb is based in Ontop which is the reason why it handles OBDA and R2RML. In the back-end component a madIS system is developed for extending a SQLite database.

5.3 Annotation models

Annotating, the act of creating associations between distinct pieces of information, is a pervasive activity online in many guises. Annotations are typically used to convey information about a resource or associations between resources. Simple examples include a comment or tag on a single web page or image, or a blog post about a news article. In this section, we present the Web Annotation Data Model, which has inspired the EOPEN annotation model described in Section 6.

[5] http://mayor2.dia.fi.upm.es/oeg-upm/index.php/en/technologies/151-geometry2rdf/

5.3.1 Web Annotation Data Model

The Web Annotation Data Model[6] specification describes a structured model and format to enable annotations to be shared and reused across different hardware and software platforms. This interoperability may be either for sharing with others, or the migration of private annotations between devices or platforms. The shared annotations must be able to be integrated into existing collections and reused without loss of significant information. Common use cases can be modelled in a manner that is simple and convenient, while at the same time enabling more complex requirements, including linking arbitrary content to a particular data point or to segments of timed multimedia resources.

The specification provides a specific JSON format for ease of creation and consumption of annotations based on the conceptual model that accommodates these use cases, and the vocabulary of terms that represents it. The Web Annotation Vocabulary[7] specifies the set of RDF classes, predicates and named entities that are used by the Web Annotation Data Model. It also lists recommended terms from other ontologies that are used in the model, and provides the JSON-LD Context and profile definitions needed to use the Web Annotation JSON serialisation in a Linked Data context.

An annotation is considered to be a set of connected resources, typically including a body and target, and conveys that the body is related to the target. The exact nature of this relationship changes according to the intention of the annotation, but the body is most frequently somehow "about" the target. This perspective results in a basic model with three parts, depicted in Figure 6. The full model supports additional functionality, enabling content to be embedded within the annotation, selecting arbitrary segments of resources, choosing the appropriate representation of a resource and providing styling hints to help clients render the annotation appropriately. Annotations created by or intended for machines are also possible, ensuring that the Data Web is not ignored in favour of only considering the human-oriented Document Web.

The Web Annotation Data Model does not prescribe a transport protocol for creating, managing and retrieving annotations. Instead it describes a resource oriented structure and serialisation of that structure that could be carried over many different protocols.

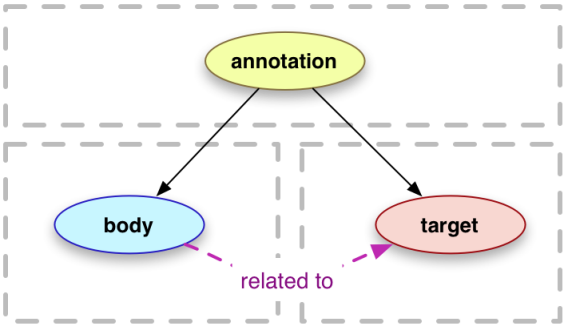

Figure 6 Core Web Annotation Data Model pattern

[6] https://www.w3.org/TR/annotation-model/

[7] https://www.w3.org/TR/annotation-vocab/

5.4 Localisation

The localisation task in EOPEN is handled by a named entity recognition system, which is responsible for detecting location entities found in tweets and subsequently pinpointing the exact tweet locations on a territory map.

During the past twenty years, since the 6th MUC conference (1995)[14], there has been enormous interest and research done in NER approaches and systems development. The main task of NER involves the detection and classification of certain types of words found in a text, such as persons, locations, organisations, temporal and numerical expressions, into respective predefined categories. Several entity type hierarchies have been introduced since then and span from simple 4-category (person, location, organisation, misc) [48] to 200-category[8] [49] propositions [45], [43].

While most first generation NER systems followed similar methodologies based on hand-crafted features, it was soon proven that the task could profit enormously by the "modus operandi" of machine learning. As such, statistical methods have been in the limelight of second generation NER approaches ever since with supervised algorithms/models like Naive Bayes, Support-Vector Machine (SVM) and especially Hidden Markov Models (HMM) and Conditional Random Fields (CRF) [26]. These systems also relied on certain hand-crafted features and were domain-dependent, requiring specific resources to be available, thus making it very difficult to adapt to a different domain. Moreover, the advent of deep learning (notably recurrent neural networks (RNN)) and a recent breakthrough in natural language understanding, word representations[9], such as word2vec [29], GloVE [37] and fastText [5], caused renewed interest in NER and led to the development of state-the-art systems, as found in [27] and [41]. A significant advantage of these latest systems is that neither feature engineering nor specialised resources are needed, other than a small amount of supervised training data, making their usage very versatile and "domain-agnostic". The latest advancements in the field take the form of models based on an encoder named Transformer [7] and deep contextualised word representations. In the latter, the trained vectors take under consideration the input sentence in its entirety and across all layers, instead of just the nearest word context of the top layer found in previous approaches. This improved results even further and newer systems are currently able to present f1-scores of around 92-93 in the Conll2003 dataset, as can be seen in ELMO [40] (92.2 F1), BERT [7] (92.8 F1), FLAIR [1] (93.09 F1) systems and in [3] (93.5 F1).

[8] https://nlp.cs.nyu.edu/ene/

[9] The idea is to represent words by vectors that contain hidden information about the treated language.

6 EOPEN ONTOLOGY AND ANNOTATION MODEL

6.1 EOPEN Annotation Model

In line with the preceding requirement analysis of EOPEN application contexts, a number of ontological constructs have been defined in order to support data modelling, integration and reasoning over the distilled information. These include:

- Constructs for capturing metadata of twitter.
- A structured model and format to enable annotations and assertions to be defined, shared and reused across both inside the EOPEN application context but also in different hardware and software platforms.

A key design choice underpinning the engineering of the EOPEN models has been the adherence to a pattern-based approach, so as to capitalise on a modular, extensible and interoperable framework for expressing annotations and achieve a better degree of knowledge sharing, reuse and interoperability. In particular, the EOPEN annotation pattern reuses the Web Annotation Data Model whose a brief summary is presented in Section 5.3.1 . It also reuses a number of existing schemata, such as DCMI and schema.org to inherit general purpose hierarchies and descriptive attributes.

It must be noted that as the modelling and reasoning requirements evolve, while also user requirements and component output become richer, iterative cycles of assessment and respective revisions will take place. These will mainly affect the domain models that we use to capture the various information types generated within EOPEN. The pattern-based approach ensures that the conceptual model for associating resources with annotations will not be affected by the updated domain models. This is especially important since it allows incremental and targeted updates to be performed on the underlying vocabularies (according to the updated requirements), minimising the risk for compatibility errors and the impact that these changes may have on the platform.

In this section, we present the way the Web Annotation Data Model is used to address the EOPEN modelling requirements, associating the tweet label information that is generated by the EOPEN modules with metadata, we call collections (Figure 7). It should be mentioned that the annotation model and underlying ontologies are checked against the requirements in order to ensure that they adequately cover the knowledge that they are expected to capture. As a consequence, formalisation and revision activities have been carried on iteratively, and will continue for the remaining duration of the project, as the use cases and requirements evolve. As already mentioned, the separation of the domain ontologies from the pattern used for attaching metadata to various resources in the form of views fosters reusability, extensibility and interoperability, minimising the effort to incorporate updates needed because of updated

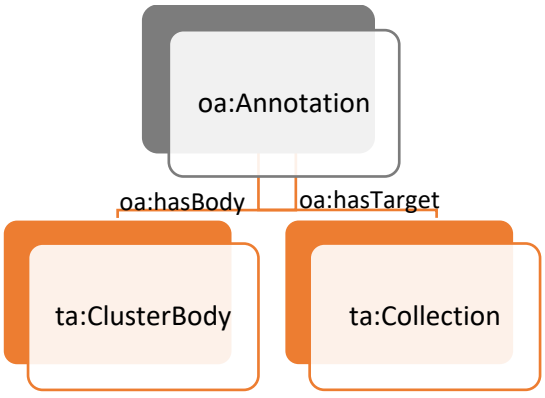

Figure 7 Core annotation model in EOPEN

user and technical requirements.

In this first version of the modelling framework (towards 1st prototype), the focus of the semantic analysis framework has been mainly given on capturing non-EO information and particularly tweet-related information and the labels that are extracted by the social media analysis modules, in line with the overall focus of the EOPEN platform. Consequently, the emphasis has been mainly placed on PUC 2. However, as described above, the standard-based annotation model we have defined can be easily extended and adapted in order to support the annotation of additional resources, without the need to change the core conceptual framework. Finally, as described in Section 6.3, the stRDF/stSPARQL and the GeoSPARQL standards have been used for EO data, which ensures the seamless support of additional EO types in subsequent versions of the framework (2nd prototype and final system).

6.1.1 EOPEN Annotation Classes

EOPEN uses the `oa:Annotation` class as a basic class in the ontology. The class is used in both `oa:Annotation` and `ta:Cluster_Annotation` instances as a root. The initiation of an annotation relevant to cluster annotation is performed by defining instances of the `oa:Annotation` class.

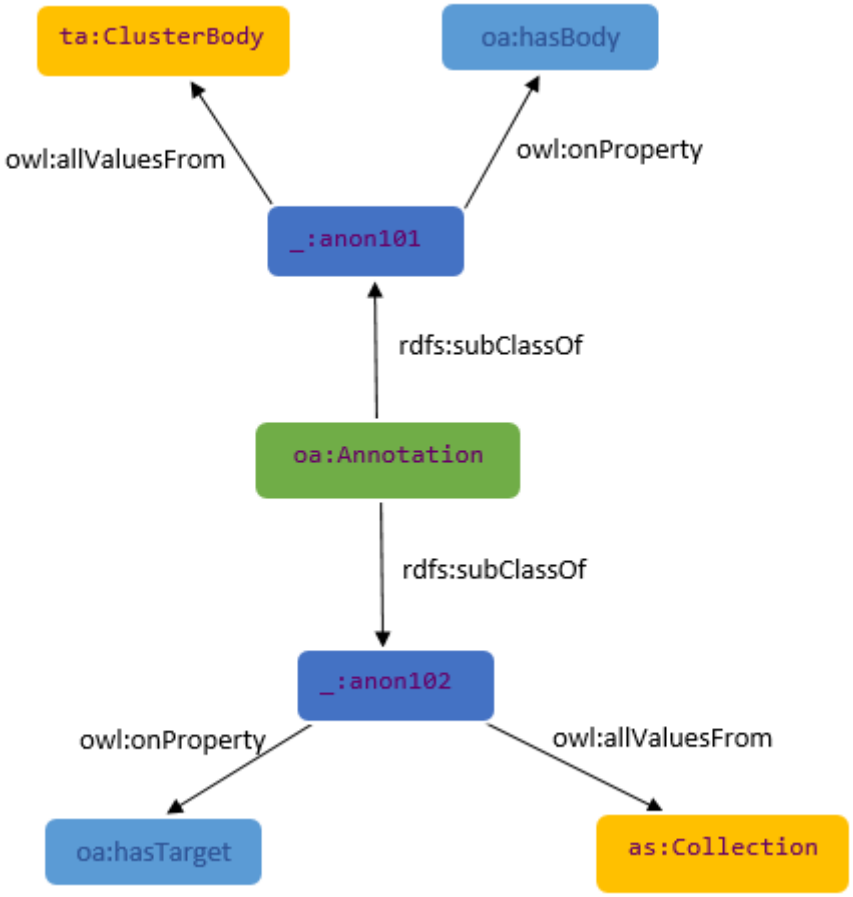

Figure 8 The base of EOPEN ontology

As depicted in Figure 8, this class restricts the `oa:hasBody` property to take as values only instances of the `ta:ClusterBody` class. In addition, the `oa:hasBody` property is restricted to take as values only instances of the `ta:Label` type, since annotations are derived only from labels. Each `ta:Label` instance has as property the name of the labels that are found in the tweet text and the accuracy of each label. The `ta:ClusterBody` is an `as:Object` type (Figure 9).

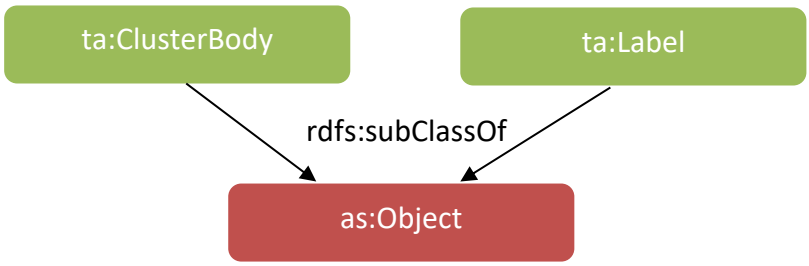

Figure 9 Classes that are subclasses of `as:Object`

The `oa:Annotation` class restricts the `oa:hasTarget` property to take as values only instances of the `ta:Collection` class (Figure 8). In addition, the `oa:hasTarget` property is restricted to take as values only instances of the `ta:Annotation` type. The annotations are restricted with the `oa:hasBody` property which take only instances of the `rdfs:Resource` class and with the `oa:hasTarget` property which take only instances of the `ta:Tweet` class. More specifically, each annotation instance is linked with a `ta:Resource` instance which contains a `ta:topRanked` property to describe if the specific tweet is characterised as top ranked. Annotation instances are also linked with a `ta:Tweet` instance which is a subclass of `as:Note` class (Figure 10).

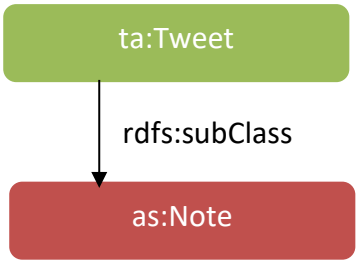

Figure 10 Class that is subclass of `as:Note`

6.2 Representing Location

The `ta:ClusterBody` instance is responsible for representing the geospatial information and information associated with the labels. For the representation of the geospatial information we have used the World Geodetic System (WGS) standard. The WGS vocabulary provides basic RDF properties for mapping geospatial data[10] like latitude and longitude. The latest version of this standard is WGS 84 (also known as WGS 1984, EPSG:4326), which was established in 1984. An example is shown below.

[10] https://www.w3.org/2003/01/geo/

Table 5 Example of World Geodetic System mapping.

```
<rdf:RDF xmlns:rdf="http://www.w3.org/1999/02/22-rdf-syntax-ns#"
         xmlns:geo="http://www.w3.org/2003/01/geo/wgs84_pos#">
  <geo:Point>
    <geo:lat>55.701</geo:lat>
    <geo:long>12.552</geo:long>
  </geo:Point>
</rdf:RDF>
```

In our case, the `ta:ClusterBody` class is associated with the WGS properties and the `ta:Label` class. Each ClusterBody refers to a coordinate system and includes a set of labels. The information that refers to the geolocation are the latitude (`wgs84_pos:lat`) and the longitude (`wgs84_pos:long`). Each label has an `as:name` property, which refers to the text of the label, and an `as:accuracy` property, which is the accuracy for the specific label. The `ta:ClusterBody` instance is associated with the following properties (Figure 11).

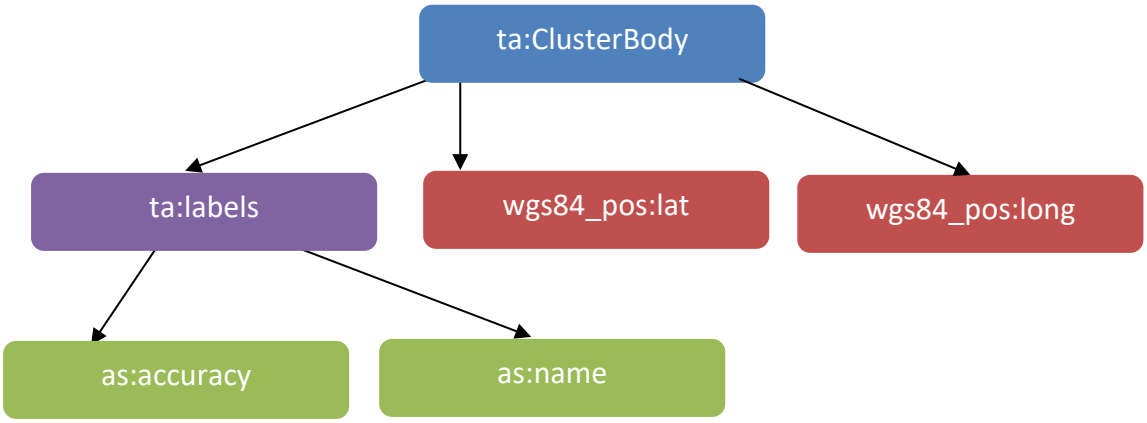

Figure 11 Properties that are associated with `ta:ClusterBody`.

6.3 Linking open EO and non-EO data

Earth Observation data in the form of satellite imagery have by their nature a spatial and a temporal dimension. Each EO product (scene) comes with an exhaustive set of structured metadata which provide the full context in which the scene has been generated. This includes information about the mission (e.g. Sentinel-1), the platform (Sentinel-1A or 1B), the sensor and the operating mode, the sensing start and end time, the orbit number, the position of the sun, etc. The information model is layered from the most generic to the most specific information. Specific information is provided for example for radar products (e.g. polarisation channels), while other information is provided for optical imagery (e.g. cloud coverage).

Similarly to tweets and events data, EO products metadata may thus be encoded as linked data and use the geo-temporal extensions of stRDF/stSPARQL or the GeoSPARQL to encode the space and time values.

An OGC Discussion Paper OGC 16-074 *EO Metadata Discovery using Linked Data*, published in March 2016[11], proposes interfaces to discover Earth Observation dataset series (families of products) and dataset (product) metadata using the Linked Data paradigm. It also describes the recommended encoding in RDF of Earth Observation resources.

Figure 12, below, extracted from the Discussion Paper, shows the main metadata attributes linked to an EO product, that is: its footprint (oml:featureOfInterest), and the sensing start and end time (ical:dtstart / ical:dtend). The document covers the full set of available metadata and provides example encodings, mainly in JSON-LD, but also RDF/XML and Turtle.

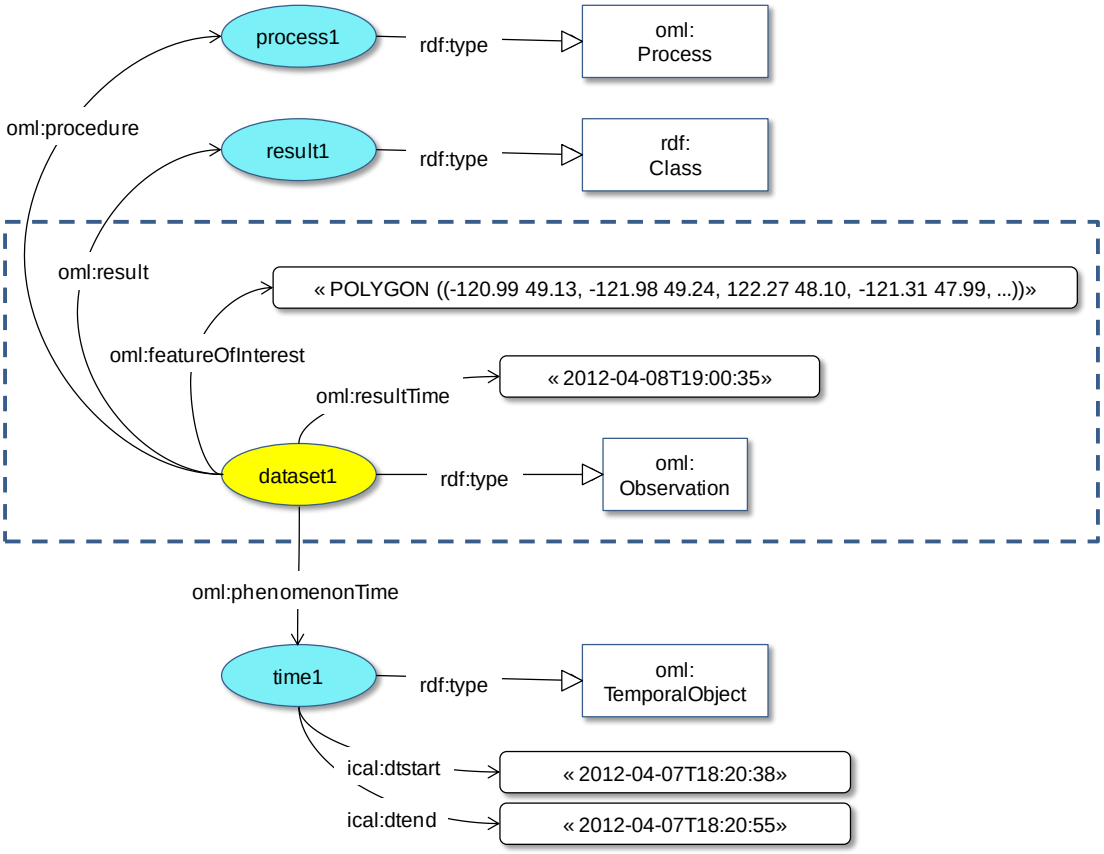

Figure 12 Earth Observation Model

The Discussion Paper leaves room for alternatives. For example, the EO product footprint could be encoded using the class oml:featureOfInterest or dct:spatial, but also sf:Polygon and the geo:asWKT relationship as specified in GeoSPARQL.

Encoding the EO products metadata using the same specifications as the tweets and the events makes it possible to store, manage and retrieve linked entities in a seamless manner, using either stSPARQL or GeoSPARQL. For example, it becomes possible to search in a single query for all the messages tweeted and all the satellite imagery acquired in the area of a given event and in the same time frame.

It also becomes possible to use an RDF browser to navigate in the tweets, events, EO products, and any other concepts that are encoded using the same classes.

[11] http://wiki.services.eoportal.org/tiki-download_wiki_attachment.php?attId=4201.

Because of the big and constantly increasing amount of available EO products, it may be difficult to systematically convert all product metadata into Linked Data and ingest the result in a graph database. Instead, a strategy may be put in place to selectively process the products, for example based on an area of interest, a time of interest, or both.

7 SEMANTIC REASONING FRAMEWORK

7.1 Localisation

In this section we present a machine learning methodology for identifying location- and organisation-type named entities in short text (tweets) written in different languages (supported languages are English, Italian and Finnish). The adopted approach for the EOPEN NER framework revolves around deep neural networks in the form of Long Short-Term Memory (LSTM)-based models and is evaluated separately with respective datasets for each treated language.

7.1.1 Methodology

In this section we present the neural network's architecture and elaborate on its most important features. Since the main issues that we need to address revolve around the candidate word's form and placement in a given sentence, we employ combined character-level and word-level representation techniques to handle them with efficacy.

As already stated, the adopted approach is based on a LSTM-based model, namely a bidirectional LSTM with a CRF layer on top. LSTMs are ideal when dealing with sequences of data (like text) because of the way the network channels information via its nodes, leveraging its ability to keep information in memory based on history. Any sequential information is being kept in the LSTM's internal state (AKA hidden layer) and is being updated with each new data via input/output and forget gates. This way the network is capable of predicting the output based on long distance dependencies. The bidirectional nature of the LSTM network manifests with two processes, applicable to each lexical unit of a given sentence that each computes a representation of the lexical unit's left and right context.

Handling word structure is not a trivial task, more so when dealing with orthographically and morphologically rich languages. To counter issues deriving from the abovementioned complexity and employ a method that takes into account the spelling of words, character-level embeddings are used. The technique involves breaking up each lexical unit in its respective characters and then feeding the resulting sequence to a bidirectional LSTM which turns it into a spelling-respecting vector.

In order to also respect the syntactic structure of a document and convey each lexical unit's contextual characteristics word-level representations are indispensable. Again, a bidirectional LSTM is used to capture each lexical unit's contextual information (both left and right context). Thus, the final word representations of the model combine both of the above embeddings.

Furthermore, to fulfil the task of NER, assigning a NER label to each word in a sentence, the output needs to be annotated accordingly. It has been shown that CRFs [26] can produce high tagging accuracy, thereby we employ a CRF layer to attribute labels for the whole sentence by leveraging sentence level tag information. The tagging format used follows the BIO scheme (B-TAG for *Beginning of entity*, I-TAG for *Inside of entity*, O-TAG for *Outside of entity*) where each word in a sentence is assigned a label reflecting its role. A named entity frequently spans not just one, but many lexical units and using this format is possible to annotate them efficiently irrespective of their length. As such, in the sentence: *Professor*

Smith tweeted that the area around London School of Economics is flooded today # London the respective annotation reads *O B-PER O O O O O B-ORG I-ORG I-ORG I-ORG O O O O B-LOC*. This tagging scheme has already yielded promising results, but in consequent testing, the format will be updated to either the BILOU or BIOES variants; they are comparable and usually improve scores even further since they predict dedicated tags for unique (U-TAG in BILOU / S-TAG in BIOES) or end entities (L-TAG in BILOU / E-TAG in BIOES). Figure 13 highlights the bi-LSTM network's structure, which is comprised of an input layer (word embeddings), a hidden layer (Bi-LSTM encoder) and an output layer (CRF layer).

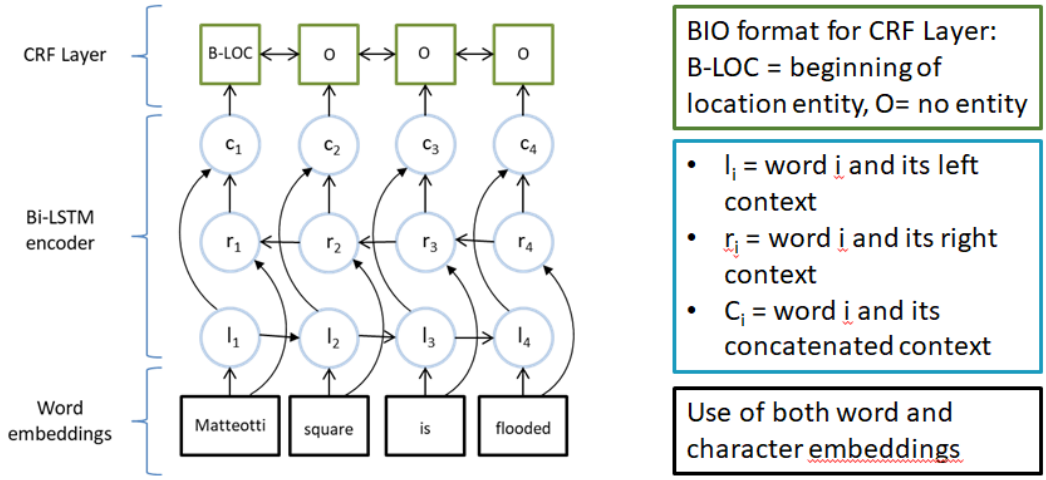

Figure 13 Bidirectional LSTM-CRF model for Named Entity Recognition

In the specific implementation data that needs to be processed (input) is comprised of two text files; the first one contains all the tweets that become available from the IR service, while the second contains the BIO annotation of the aforementioned tweets, one tweet per line for both. The results are then outputted in a single file, in a "one sentence token-per-line" format. In the same line, apart from the sentence token the assigned label is also displayed in a tab delimited format.

The NER process using our model is followed by the concatenation of the sequential entities recognised as location or organisation, unless there is a comma or a full stop. Eventually, two lists are produced that contain all the locations and the organisations recognised within the sentence, respectively (Figure 14).

In the case where location entities are recognised, the bounding box[12] of each location is retrieved via the OpenStreetMap API. It should be noted that usually OpenStreetMap returns multiple responses for an entity and lists them based on their popularity. We consider as correct the most popular one and retrieve the coordinates of its bounding box and the coordinates of a specific point. In case no location entities are recognised, the organisation entities are considered. Otherwise, the output of the tweet localisation module is Null.

Finally, an analysis of the bounding boxes returned follows. Specifically, in case of one entity, a single bounding box is returned. However, in case of multiple entities, the bounding boxes

[12] https://wiki.openstreetmap.org/wiki/API_v0.6

are compared with each other in order to exclude bigger areas when a smaller - more precise one is also available and all the remaining are returned as output.

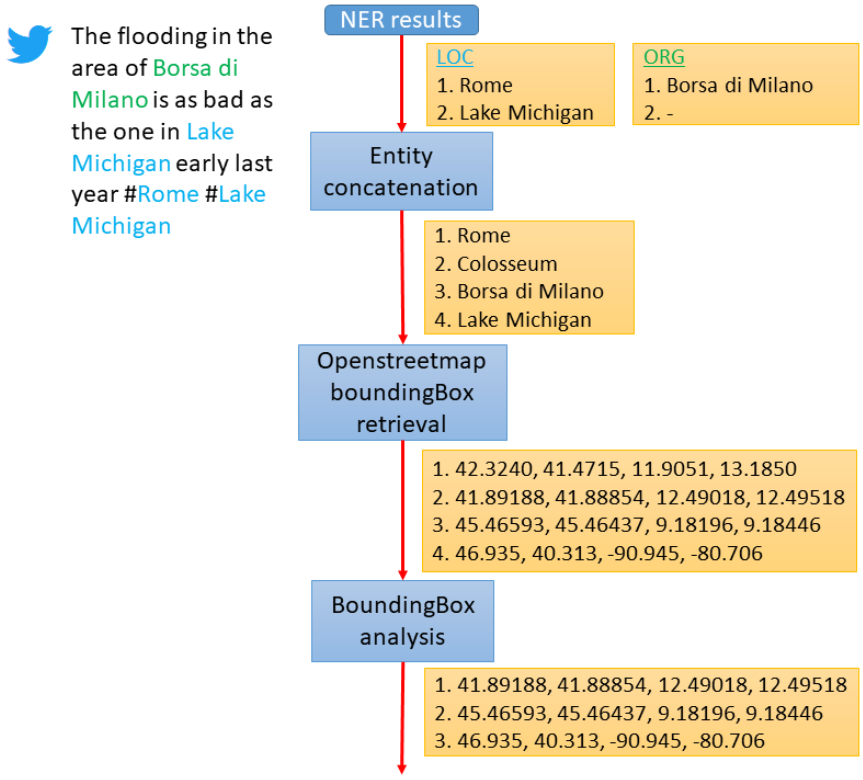

Figure 14 Localisation steps after NER has been performed on available tweets

7.1.2 **Datasets**

A make or break feature concerning machine learning approaches is the availability of appropriate datasets that will be used to train the relevant computational models. Such datasets are easily attainable when over-represented languages like English are involved. However, the same does not apply to less represented languages like Italian or Finnish; the resources are scarce and finding a publicly available dataset can pose an insurmountable problem. Moreover, even when a suitable dataset is encountered, it is often in unconventional format or/and optimised for traditional classifier methods and as such, not adequate for training neural network-based models, which currently hold the state-of-the-art crown. There are also cases where training data do exist but are limited in size. Thus, a lot of effort must be put into converting them to secure compatibility with contemporary models or into fusing them with other also limited datasets. To address the specific issue, many researchers are forced to create datasets from scratch, a process which is both expensive and time-consuming. In the context of EOPEN both publicly available dataset and custom annotated corpora are exploited to manage multilingual location and organisation extraction entities.

In the context of the English localisation task, the model has been trained and evaluated on the CoNNL2003 dataset [48] which includes annotation of 4 categories of named entities: i) Person (PER), ii) Location (LOC), iii) Organisation (ORG), and iv) Miscellaneous (MISC). Out of these annotations, we have direct interest only for the “Location” one, but also opted for the “Organisation” entities in an attempt to infer possible locations out of organisations’ locales.

The task will be handled by a separate procedure that will consider an organisation's name being present in a tweet along with respective context analysis which will include heuristics concerning phrases like "adjacent/next to", "2 kms", "50 metres", etc.

The dataset is balanced in the three important categories (PER, ORG, LOC) and contains similar number of named entity occurrences, as is evident from Table 6.

Table 6 Number of named entities and sentences in each of the CoNLL2003 data files

CoNLL2003 (EN)	PER	LOC	ORG	MISC	Sentences
Training set	6600	7140	6321	3438	14987
Development set	1842	1837	1341	922	3466
Test set	1617	1668	1661	702	3684

Concerning the Italian task, the dataset originates from the NER task at EVALITA 2009 [51]. The tagging method that was followed is based on the IOB2 format while the entity types differ in respect to the CoNLL2003 dataset in that the MISC category is replaced by the GPE (Geo-Political Entity) one.

Table 7 Number of named entities and sentences in each of the EVALITA2009 data files.

EVALITA2009 (IT)	PER	LOC	ORG	GPE	Sentences
Training set	4577	362	3658	2813	11227
Test set	2378	156	1289	1143	4136

When dealing with NER in Finnish, the issue of limited language resources quickly becomes apparent. To the best of our knowledge there is only one freely available dataset[13] which was extracted from the archives of Digitoday, a Finnish online technology news source. It consists of 953 articles in BIO format, annotated with six named entity classes (organisation, location, person, product (PRO), event (EVENT), and date (DATE)). Since the dataset in its current state is too small to be used with a DNN, no remarkable results are expected until the issue of data size is addressed. On that account, we are working towards enhancing it, by adding more sentences of our own manual annotation efforts.

Table 8 Number of named entities and sentences in each of the DIGITODAY data files

DIGITODAY (FI)	PER	LOC	ORG	PRO	EVENT	DATE	Sentences
Training set	84	58	519	135	1	78	591
Test set	185	268	544	173	2	51	986

7.1.3 **Network parameters and training**

The parameters that were used during training of the Bi-LSTM-CRF models are displayed on Table 9. The word representations used as input in the current model are the publicly available, pre-trained 300-dimensional GloVE embeddings [37] trained on the common crawl

[13] https://github.com/mpsilfve/finer-data

corpus for the English language, while for Italian and Finnish, the respective pre-trained 300-dimensional fastText [13] embeddings were used. At the time of writing, the same settings were applied to all three languages. However, in the coming months they will be fine-tuned to perform optimally, taking advantage of all differentiating features of the specific resources.

Table 9 Bi-LSTM-CRF settings used for the task's languages

Training parameters	Value
Optimiser	Adam
Character embeddings dimensions	100
Word embeddings dimensions	300
Dropout rate	0.5
Epochs	25
Batch size	20
LSTM size	100

7.1.4 **Results**

For the evaluation of the performance of all configurations the values for the precision, recall and F1-score measures were computed (Table 10). It should be noted at this point that all reported results concern the entirety of named entities found in the respective datasets, while our actual area of interest is limited only to the location entities found in tweets. Dedicated location scores, comparison with other systems where available, and appropriate fine-tuning information will be reported on D5.2.

Table 10 Performance evaluation of the proposed system (EN) vs. the baseline system and a state-of-the-art approach

System (CoNLL2003)	Precision	Recall	F1-score
Our system	90.95	90.94	90.97
Best-scoring shared task system: Florian et al., 2003	88.99	88.54	88.76
Baevski, A. et al. 2019	(not reported)	(not reported)	93.5

At this point the results concerning the English language are very similar to the state-of-the-art, yielding an F1-score of ~91. To further improve on this score, on future iterations we plan to update the model with newer embeddings (e.g. ELMO) and annotation (to the BILOU/BIOES format). Next, we introduce ten example tweets extracted directly from the EOPEN platform after they have been processed by the localisation tool. The sentences are presented with integrated location annotation to better convey the application process. In orange the entities that were extracted correctly, while in red the entities that were not extracted at all or were extracted with the wrong annotation (e.g. PER instead of LOC) and

thus, discarded. All coloured entities in each sentence should be extracted and annotated correctly in order to reach 100% results.

Relevant passage (EN):

1. Matteotti square is flooded. #underwater #flooding
2. The sewers are flooded. #Vicenza #flooding
3. #Bacchiglione #flooding #Vicenza The river has overflowed.
4. The levees are cracked at Angeli bridge.
5. Houston fears climate change will cause catastrophic flooding
6. Could see heavy rain and local flooding from storms on Monday in New Jersey …
7. How quick-thinking mother saved family from Grenfell fire by flooding her flat
8. Flying in over the snow covered fields of Finland was quite magical!
9. current weather in Tampere: light shower snow, -4°C, 92% humidity, wind 3kmh
10. Not only one, many snowploughs coming to the rescue. #oslo

The same evaluation metrics apply to the Italian use case and are presented in Table 11.

Table 11 Performance evaluation of the proposed system (IT) vs. the baseline system and a state-of-the-art approach

System (EVALITA2009)	Precision	Recall	F1-score
Our system	75.49	75.60	75.37
Best-scoring shared task system: FBK_ZanoliPianta	84.07	80.02	82.00
Nguyen and Moschitti, 2012	85.99	82.73	84.33

Currently the Italian model is fully operational and the first results are indicative of its ongoing development status. The dataset needs to be updated in terms of annotation (to the BILOU/BIOES format) and the training parameters and embeddings need to be fine-tuned in order to take advantage of the neural model's potential. The best result to date is reported by such a system, as can be seen in [33]. Again, as was the case with the English language, ten example tweets are presented to demonstrate the tool's progress towards efficient location recognition in Italian.

Relevant passage (IT):

1. Ventennale dell'#alluvione di #Sarno, cosa è cambiato?
2. Dicono che Genova è solo rossoblù e fanno il tifo per l'alluvione
3. esiste tifoseria più ritardata del Napoli?
4. Presentazione il sistema di #allertameteo della #ProtezioneCivile della città di #Gorizia
5. Situazione di forte #allertameteo ieri in #Spagna per la #grandine.
6. #Siracusa, allagamento nel seminterrato dell'ospedale "Rizza"
7. Siamo al 21° anniversario dell'alluvione a Stazzema: una targa in ricordo
8. #documentario sull'alluvione #Firenze al @AquaFilmFestiva 2017
9. Maltempo, disastro in Veneto: dopo il super caldo, ecco l'alluvione.

10. Ponte Milvio fa acqua: ancora un allagamento in via Prati della Farnesina... #news #Roma

7.2 Semantic search enrichment

In this section we present a methodology that we developed to extract the first hypernyms in three levels from WordNet. The methodology establishes a connection between BabelNet, Babelfy and WordNet. More specifically the methodology gets as input a JSON file which contains mostly text metadata about a topic which is found in tweets like labels, location and ids Table 12. In the JSONArray each JSONElement characterises a different topic. More details about the JSON input file can be found in section 8.

Table 12 Input example


```
[
   {
      "id":"1",
      "labels":[
         {
            "text":"lash",
            "score":0.0339359955414791
         },
         {
            "text":"breaking",
            "score":0.0257992605776155
         },
         {
            "text":"area",
            "score":1.1073852024257E-7
         }
      ],
      "location":{
         "latitude":45.5455,
         "longitude":11.5354
      },
      "tweets":[
         "1001376987271778310",
         "1001379246076379136",
         "1001379428713119745",
         "1001379709177942016",
         "1001412744237629446",
         "1001411514874585088"
      ],
      "top_ranked_tweets":[
         "1001376987271778310",
         "1001379246076379136",
         "1001379428713119745",
         "1001379709177942016"
      ]
   }
]
```


We take into account each label's text from the input (Value selection) and we extract Babelfy information for each text using the Babelfy Java API (Information extraction). Some words may appear multiple times in the response, so we apply a data cleaning technique to

keep the words with the highest global score which is included in the Babelfy information that are extracted (Results cleaning). Another field that is included is the BabelNet URL, which we used to connect to the BabelNet Linked data interface[14] (WordNet id extraction). The interface contains a plethora of additional information for each word. We used the lemon-WordNet31 value to execute SPARQL queries and extract the first hypernym for each word for a three-level tree of hypernyms (Table 13). A graph view of the methodology that we followed is shown in Figure 15.

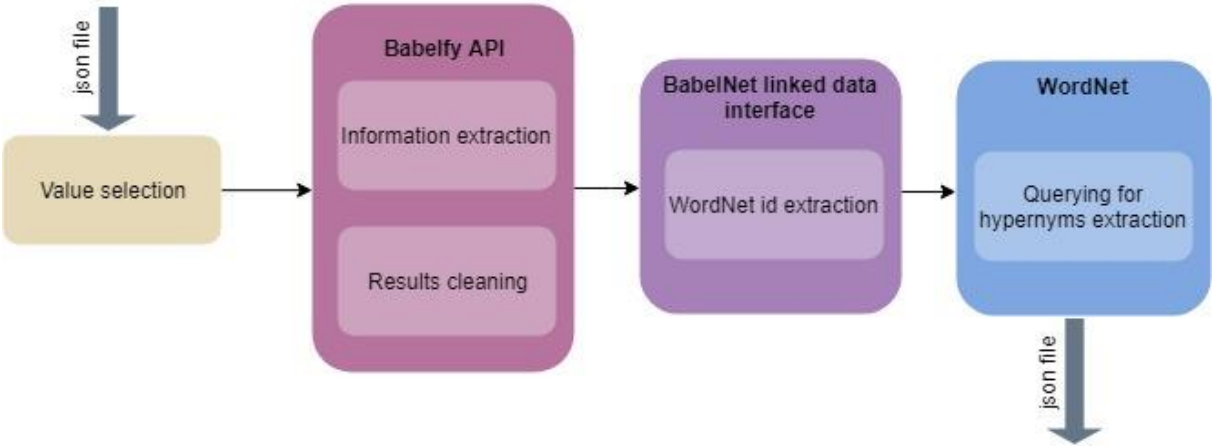

Figure 15 The methodology that we followed

The results are saved in a new JSON file which contains the plain text and the tags from the original JSON file, the file id and a JSON array with the WordNet information: the term that has been analysed and the hypernym identifier for each one of the three hypernym levels Table 13. Some results were not found in BabelNet and they are not included in the result, while others may not have hypernyms so they have "null" value in the hypernym level that was not found. The identifier of each level can be used in WordNet to extract more information about the each hypernym like the description, the synonyms, the part of speech type, the hypernyms, the hyponyms and the holonyms.

Table 13 Example of querying in WordNet for hypernyms extraction

```
PREFIX pwnid: <http://wordnet-rdf.princeton.edu/id/>
PREFIX wn: <http://wordnet-rdf.princeton.edu/ontology#>
PREFIX ontolex: <http://www.w3.org/ns/lemon/ontolex#>

select distinct ?x where {
        pwnid:05135784-n  wn:hypernym ?x.
}
```

[14] http://babelnet.org/rdf/page/

Table 14 Output example


```
[
   {
      "Id":"1",
      "Wordnet":[
         {
            "Level 1":"01881348-v",
            "Level 3":"01835473-v",
            "Level 2":"01880523-v",
            "Term":"lash"
         },
         {
            "Level 1":"null",
            "Level 3":"null",
            "Level 2":"null",
            "Term":"breaking"
         },
         {
            "Level 1":"05130681-n",
            "Level 3":"04923519-n",
            "Level 2":"05097645-n",
            "Term":"area"
         }
      ]
   }
]
```


7.3 Rules

The semantic framework implements a reasoning and validation framework to further aggregate and link the metadata collected from processing components, as well as to ensure the semantic consistency of the generated RDF graphs.

Additional inferences are derived by combining native OWL 2 RL reasoning and custom rule sets. The former is solely based on the OWL 2 RL profile semantics that are implemented in the form of first-order axiomatisation (OWL 2 RL/RDF rules [30]). The knowledge base is connected with state-of-the-art triple stores, such as GraphDB and AllegroGraph, for storing and querying metadata that inherently implement the semantics of OWL 2 RL.

However, the semantics of the OWL 2 language is limited. For example, OWL 2 class semantics can model only domains where instances are connected in a tree-like manner [32]. Our framework allows us to define custom rules on top of the knowledge graphs to express richer semantic relations. SPARQL-based rules are used to capture domain relationships in the form of CONSTRUCT graph patterns that identify the valid inferences that can be made on the annotation pattern. Figure 16 presents the abstract reasoning architecture.

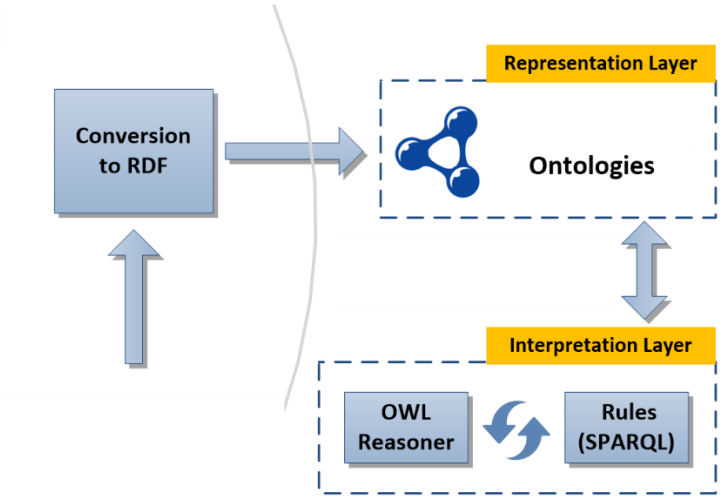

Figure 16 Abstract reasoning architecture

7.3.1 **Rules**

We use SPIN rules, i.e. SPARQL construct graph patterns, to implement expressive reasoning rules, enabling property value propagation and instance generation (when needed). The core idea is to associate each reasoning task with one or more SPARQL rules that support specific reasoning functionality, e.g. link of tweet clusters. In the following, we present examples of such reasoning cases and rules. More elaborate rule-based reasoning cases will be tackled in future versions of the prototype framework and reported in upcoming deliverables.

Tweet cluster linking. In section 7.2 we described the way tweet clusters are enriched with additional terms from lexical database and Linked Data resources. Based on the new context added to the KB, we can define rules that materialise and further interlink the resources. For example, tweet cluster whose extended list of topics contains many overlaps with another tweet cluster, can be considered relevant. In this case, we enable the navigation among of conceptually relevant tweet groups, detecting connections that have not been identified by the topic detection module. The rule that implements this relations is given below:

```
PREFIX ta: <https://eopen-project.eu/ontologies/tweet-annotations#>
PREFIX oa: <http://www.w3.org/ns/oa#>
CONSTRUCT {
    ?a1 ta:relevant ?a2 .
}
WHERE {
    ?a1 a ta:ClusterAnnotation;
       oa:hasBody ?b1.
    ?b1 ta:labels ?l1.
    ?a2 a ta:ClusterAnnotation;
       oa:hasBody ?b2.
    ?b2 ta:labels ?l2.
    FILTER (?a1 != ?a2) .
    FILTER(:relevant (?l1, ?l2))
}
```

The rule matches a tweet annotation (a1) whose cluster contains at least one relevant topic to some other cluster (a2). In this case, a new triple is generated that materialise the fact that these two cluster are relevant (`?a1 ta:relevant ?a2`). In future version of the framework, we will implement a more advanced method that will take into account the number of the relevant terms in order to associate different weights.

7.3.2 **Validation**

The validation process aims to check the consistency, structural and syntactic quality of the generated metadata descriptions before the annotations graphs are persisted in the repository. The validity of the annotations is checked using both native ontology consistency checking (e.g. OWL 2 DL reasoning) and custom SHACL [22] validation rules, following the closed-world paradigm. The former handles validation taking into account the semantics at the terminological level (TBox), e.g. checking class disjointness. The latter is necessary in order to detect constraint violations in RDF data, e.g. missing values, cardinality violations, etc. An example SHACL shape is given below that represents a constraint that all Cluster Annotations should include at least one collection of tweets.

```
ta:ClusterAnnotation
   rdf:type sh:NodeShape ;
   sh:property [
      rdf:type sh:PropertyShape ;
      sh:path oa:hasTarget ;
      sh:class as:Collection ;
      sh:minCount 1 ;
      sh:nodeKind sh:IRI ;
   ] ;
.
```

8 ONTOLOGY VALIDATION

In this section we describe the EOPEN annotation model to map the results of Task 4.2 using a simulation example. The technical partners provided feedback about the generated results (both in terms of the format and content), helping us generate the respective annotation vocabularies. The JSON below was given as input and the Turtle RDF file was produced as an output.

Table 15 Example of input data in JSON


```
[
   {
      "id":"1",
      "labels":[
         {
            "text":"rain",
            "score":0.320496732575757
         }
        ],
       "location":{
          "latitude":45.5455,
          "longitude":11.5354
      },
      "tweets":[
        "1001376594013765632"
      ],
      "top_ranked_tweets":[
         "1001376594013765632"
      ]
   }
]
```


The input describes the results of topic extraction. The results contain a top ranked tweet which has one label. The tweet refers to a geolocation. The output contains properties relevant with the resource and labels found in the tweet.

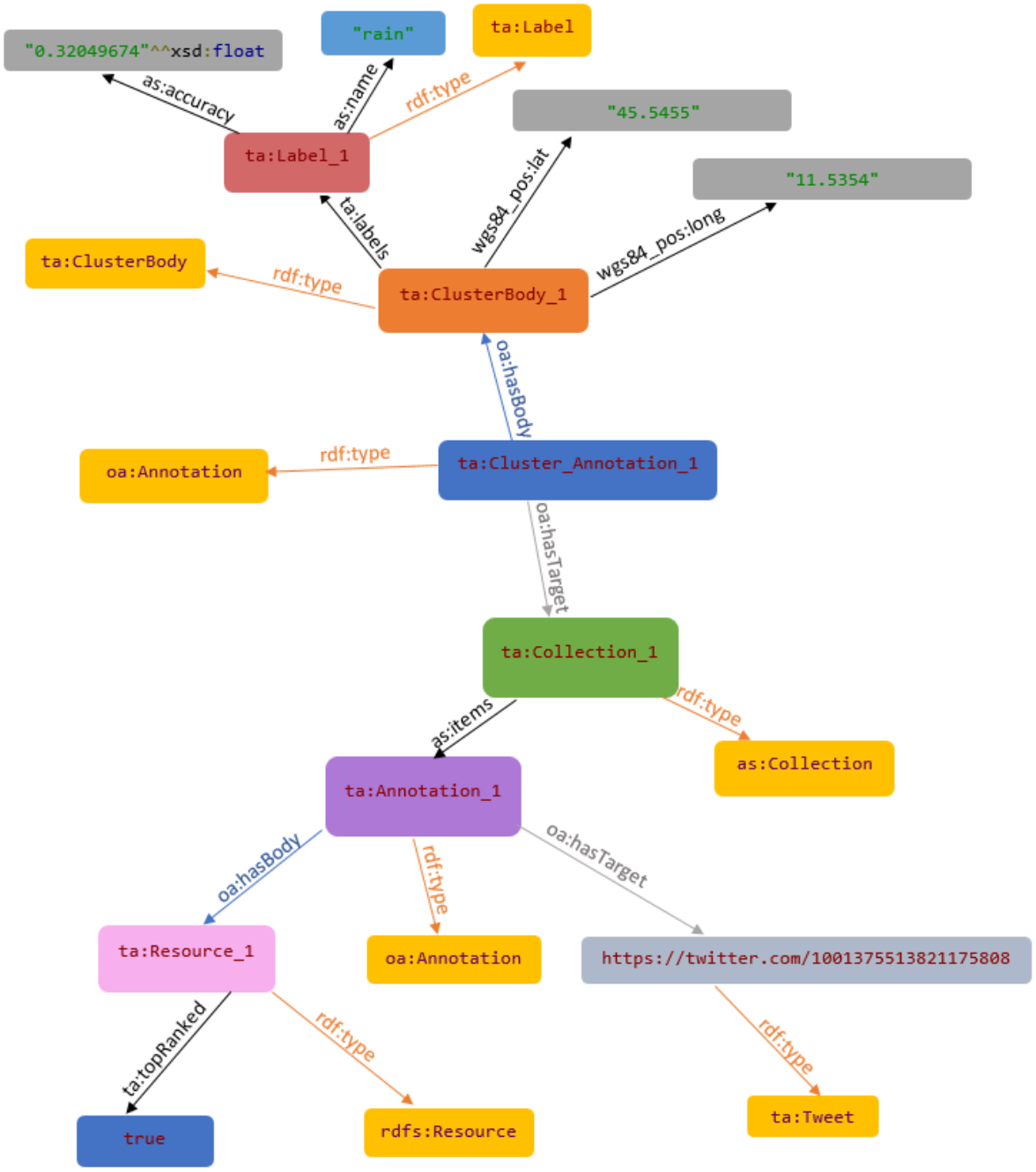

Figure 17 Example mapping of topic extraction results for the simulation example

The generated knowledge graphs contain all the necessary relations to adequately map the output of topic extraction. Following the annotation model described in Section 6.1 , an `ta:ClusterAnnotation` resource is generated that is linked with the target of the annotation, i.e. the generated collection (`Collection_1`) and the cluster body (`ta:ClusterBody_1`). The latter, defines property assertions relevant to the labels extracted from the tweet, the geolocation and the resource of the tweet, while the first defines the tweet general information like URL and topRanked property. The RDF graph in the Turtle syntax[15] is given below.

[15] https://www.w3.org/TR/turtle/

Table 16 Example of output data in RDF

```
@prefix owl: <http://www.w3.org/2002/07/owl#> .
@prefix rdfs: <http://www.w3.org/2000/01/rdf-schema#> .
@prefix xsd: <http://www.w3.org/2001/XMLSchema#> .
@prefix oa: <http://www.w3.org/ns/oa#> .
@prefix as: <http://www.w3.org/ns/activitystreams#> .
@prefix ta: <https://eopen-project.eu/ontologies/tweet-annotations#> .
@prefix wgs84_pos: <http://www.w3.org/2003/01/geo/wgs84_pos#> .

<https://eopen-project.eu/ontologies/tweet-annotations#>
  a owl:Ontology ;
  owl:imports <http://www.w3.org/ns/oa>, <https://www.w3.org/ns/activitystreams-
owl> ;
  owl:versionInfo "Created with TopBraid Composer" .

<https://eopen-project.eu/ontologies/tweet-annotations#ClusterBody>
  a owl:Class ;
  rdfs:subClassOf <http://www.w3.org/ns/activitystreams#Object> .

<https://eopen-project.eu/ontologies/tweet-annotations#Label>
  a owl:Class ;
  rdfs:subClassOf <http://www.w3.org/ns/activitystreams#Object> .

<https://eopen-project.eu/ontologies/tweet-annotations#Tweet>
  a owl:Class ;
  rdfs:subClassOf <http://www.w3.org/ns/activitystreams#Note> .

<https://eopen-project.eu/ontologies/tweet-annotations#labels>
  a owl:ObjectProperty ;
  rdfs:domain <https://eopen-project.eu/ontologies/tweet-annotations#ClusterBody>
;
  rdfs:range <https://eopen-project.eu/ontologies/tweet-annotations#Label> ;
  rdfs:subPropertyOf <http://www.w3.org/ns/activitystreams#tags> .

<https://eopen-project.eu/ontologies/tweet-annotations#topRanked>
  a owl:DatatypeProperty ;
  rdfs:range xsd:boolean .

<https://eopen-project.eu/ontologies/tweet-annotations#Cluster_Annotation_1>
  a <http://www.w3.org/ns/oa#Annotation> ;
  oa:hasBody <https://eopen-project.eu/ontologies/tweet-annotations#ClusterBody_1>
;
  oa:hasTarget <https://eopen-project.eu/ontologies/tweet-
annotations#Collection_1> .

<https://eopen-project.eu/ontologies/tweet-annotations#Label_1>
  a <https://eopen-project.eu/ontologies/tweet-annotations#Label> ;
  as:accuracy "0.32049674"^^xsd:float ;
  as:name "rain" .

<https://eopen-project.eu/ontologies/tweet-annotations#Annotation_1>
  a oa:Annotation ;
  oa:hasBody <https://eopen-project.eu/ontologies/tweet-annotations#Resource_1> ;
  oa:hasTarget <https://twitter.com/1001376594013765632> .
```

```
<https://twitter.com/1001376594013765632> a <https://eopen-
project.eu/ontologies/tweet-annotations#Tweet> .
<https://eopen-project.eu/ontologies/tweet-annotations#Resource_1>
  a rdfs:Resource ;
  ta:topRanked true .

ta:Collection_1
  a as:Collection ;
  as:items ta:Annotation_1 .

ta:ClusterBody_1
  a ta:ClusterBody ;
  wgs84_pos:lat "45.5455"^^xsd:float ;
  wgs84_pos:long "11.5354"^^xsd:float ;
  ta:labels ta:Label_1 .
```

9 CONCLUSIONS

In this document we provided the requirement specifications and the state-of-the-art analysis relevant to the building of the semantic knowledge structures addressed within T5.1 "The EOPEN ontology". We also described the current status of the EOPEN ontologies towards MS3 that encode in a structured way the vocabulary and the precise semantics of information relevant to the EOPEN application context. We have also presented the preliminary version of WP5's reasoning framework towards MS3 (T5.2 "Reasoning for decision support") for combining, integrating and semantically interpreting and enriching the knowledge captured in the KB. The current annotation model of EOPEN has been validated through a simulation example organised within WP5 in order to elicit modelling requirements and acquire a better understanding of the structure and content of the outputs provided by each component of the EOPEN pipeline. Finally, as far as T5.2 is concerned ("Linked open EO data"), we presented the roadmap towards interlinking EO and non-EO data.

Next steps include further enrichments and enhancements of WP5 ontology-based framework in three main directions. First, to refine the already developed annotation models and to provide and validate additional ontology constructs for capturing richer domain knowledge pertinent population of the KB with data, based on the richer output the various modules will provide towards the first prototype (M18). This involves also the development and integration of domain models pertinent to the use cases of the EOPEN project. The annotation model will be also enriched with additional metadata properties, when it is a clear view on the exact output of the analysis. Second, to enhance the reasoning capabilities that will address more elaborate interpretation aspects by (i) enriching the supported semantics both at the terminological level, by defining additional class and property axioms, and at the assertional level by incorporating inference rules, (ii) handling imperfect information (i.e. missing or uncertain inputs). Special emphasis will be also place on aggregating the results of EO and non-EO data. Finally, in parallel with T5.2, searching APIs will be implemented in order to provide an intelligent query interface for addressing users' searching requirements, e.g. searching tweets based on semantics.

REFERENCES

[1] Akbik, A. et al. 2018. Contextual string embeddings for sequence labeling. *Proceedings of the 27th International Conference on Computational Linguistics* (2018), 1638–1649.

[2] Baader, F. et al. 2003. *The description logic handbook: Theory, implementation and applications*. Cambridge university press.

[3] Baevski, A. et al. 2019. Cloze-driven Pretraining of Self-attention Networks. *arXiv preprint arXiv:1903.07785*. (2019).

[4] Bereta, K. et al. 2018. Querying the web on-the-fly using ontologies and mappings. *CEUR Workshop Proceedings* (2018).

[5] Bojanowski, P. et al. 2016. Enriching Word Vectors with Subword Information. *arXiv preprint arXiv:1607.04606*. (2016).

[6] Deborah L. McGuinness, F. van H. 2004. Owl web ontology language overview. *W3C recommendation 10.2004-03*. (2004). DOI:https://doi.org/10.1145/1295289.1295290.

[7] Devlin, J. et al. 2018. Bert: Pre-training of deep bidirectional transformers for language understanding. *arXiv preprint arXiv:1810.04805*. (2018).

[8] Dimou, A. et al. 2015. Assessing and refining mappings to RDF to improve dataset quality. *Lecture Notes in Computer Science (including subseries Lecture Notes in Artificial Intelligence and Lecture Notes in Bioinformatics)* (2015).

[9] Dimou, A. et al. 2014. RML: A generic language for integrated RDF mappings of heterogeneous data. *CEUR Workshop Proceedings* (2014).

[10] Fernández-López, M. et al. 1997. METHONTOLOGY: From Ontological Art Towards Ontological Engineering. *AAAI-97 Spring Symposium Series*. (1997). DOI:https://doi.org/10.1109/AXMEDIS.2007.19.

[11] Glimm, B. et al. 2014. HermiT: An OWL 2 Reasoner. *Journal of Automated Reasoning*. (2014). DOI:https://doi.org/10.1007/s10817-014-9305-1.

[12] Grau, B.C. et al. 2008. OWL 2: The next step for OWL. *Web Semantics: Science, Services and Agents on the World Wide Web*. 6, 4 (2008), 309–322.

[13] Grave, E. et al. 2018. Learning Word Vectors for 157 Languages. *Proceedings of the International Conference on Language Resources and Evaluation (LREC 2018)* (2018).

[14] Grishman, R. and Sundheim, B. 1996. Message understanding conference-6: A brief history. *COLING 1996 Volume 1: The 16th International Conference on Computational Linguistics* (1996).

[15] Grosof, B.N. et al. 2003. Description logic programs: Combining logic programs with description logic. *Proceedings of the 12th international conference on World Wide Web*. (2003). DOI:https://doi.org/10.2139/ssrn.460986.

[16] Gruber, T.R. 1993. A translation approach to portable ontology specifications. *Knowledge Acquisition*. (1993). DOI:https://doi.org/10.1006/knac.1993.1008.

[17] Haarslev, V. and Möller, R. 2003. Racer: A Core Inference Engine for the Semantic Web. *Proceedings of the 2nd International Workshop on Evaluation of Ontologybased*

Tools. (2003). DOI:https://doi.org/10.1.1.10.3822.

[18] Harris, S. and Seaborne, A. 2013. *SPARQL 1.1 Query Language*.

[19] He, S.Y. et al. 2004. Effects of pres0sure reduction rate on quality and ultrastructure of iceberg lettuce after vacuum cooling and storage. *Postharvest Biology and Technology* (2004), 263–273.

[20] Horrocks, I. et al. 2004. SWRL : A Semantic Web Rule Language Combining OWL and RuleML. *W3C Member submission 21*. (2004).

[21] ter Horst, H.J. 2004. Extending the RDFS Entailment Lemma. *The Semantic Web – ISWC 2004* (2004).

[22] Knublauch, H. and Ryman, A. 2015. Shapes Constraint Language (SHACL). W3C First Public Working Draft, W3C.

[23] Koubarakis, M. and Kyzirakos, K. 2010. Modeling and querying metadata in the semantic sensor web: The model stRDF and the query language stSPARQL. *Lecture Notes in Computer Science (including subseries Lecture Notes in Artificial Intelligence and Lecture Notes in Bioinformatics)* (2010).

[24] Kyzirakos, K. et al. 2018. GeoTriples: Transforming geospatial data into RDF graphs using R2RML and RML mappings. *Journal of Web Semantics*. (2018). DOI:https://doi.org/10.1016/j.websem.2018.08.003.

[25] Kyzirakos, K. et al. 2012. Strabon: A semantic geospatial DBMS. *Lecture Notes in Computer Science (including subseries Lecture Notes in Artificial Intelligence and Lecture Notes in Bioinformatics)* (2012).

[26] Lafferty, J. et al. 2001. Conditional random fields: Probabilistic models for segmenting and labeling sequence data. (2001).

[27] Lample, G. et al. 2016. Neural architectures for named entity recognition. *arXiv preprint arXiv:1603.01360*. (2016).

[28] León, A. De et al. 2010. Geographical linked data: a Spanish use case. *Proceedings of the In I-SEMANTICS '10 6th International Conference on Semantic Systems* (2010).

[29] Mikolov, T. et al. 2013. Efficient estimation of word representations in vector space. *arXiv preprint arXiv:1301.3781*. (2013).

[30] Motik, B. et al. 2012. OWL 2 Web Ontology Language Profiles (Second Edition). *W3C Recommendation*.

[31] Motik, B. et al. 2005. Query answering for OWL-DL with rules. *Web Semantics*. (2005). DOI:https://doi.org/10.1016/j.websem.2005.05.001.

[32] Motik, B. et al. 2008. Structured objects in OWL: Representation and reasoning. *Proceedings of the 17th international conference on World Wide Web* (2008), 555–564.

[33] Nguyen, T.-V.T. and Moschitti, A. 2012. Structural reranking models for named entity recognition. *Intelligenza Artificiale*. 6, 2 (2012), 177–190.

[34] De Nicola, A. et al. 2005. A Proposal for a Unified Process for Ontology Building:

UPON. Springer, Berlin, Heidelberg. 655–664.

[35] Noy, N.F. and McGuinness, D.L. 2001. Ontology Development 101: A Guide to Creating Your First Ontology. *Stanford Knowledge Systems Laboratory*. (2001). DOI:https://doi.org/10.1016/j.artmed.2004.01.014.

[36] Patroumpas, K. et al. 2014. TripleGeo: An ETL tool for transforming geospatial data into RDF triples. *CEUR Workshop Proceedings* (2014).

[37] Pennington, J. et al. 2014. Glove: Global vectors for word representation. *Proceedings of the 2014 conference on empirical methods in natural language processing (EMNLP)* (2014), 1532–1543.

[38] Perez, J. et al. 2006. Semantics and Complexity of SPARQL. (2006), 30–43. DOI:https://doi.org/10.1145/1567274.1567278.

[39] Perry, M. and Herring, J. 2012. OGC GeoSPARQL-A geographic query language for RDF data. *OGC Candidate Implementation Standard*. (2012).

[40] Peters, M.E. et al. 2018. Deep contextualized word representations. *arXiv preprint arXiv:1802.05365*. (2018).

[41] Peters, M.E. et al. 2017. Semi-supervised sequence tagging with bidirectional language models. *arXiv preprint arXiv:1705.00108*. (2017).

[42] Pittaras, N. et al. 2019. GeoSensor: semantifying change and event detection over big data. *Proceedings of the 34th ACM/SIGAPP Symposium on Applied Computing* (2019), 2259–2266.

[43] Pradhan, S. et al. 2013. Towards robust linguistic analysis using OntoNotes. *Proceedings of the Seventeenth Conference on Computational Natural Language Learning* (2013), 143–152.

[44] R2RM: RDB to RDF Mapping Language: 2011. .

[45] Ritter, A. et al. 2011. Named entity recognition in tweets: an experimental study. *Proceedings of the conference on empirical methods in natural language processing* (2011), 1524–1534.

[46] Rosati, R. 2006. DL+log: Tight Integration of Description Logics and Disjunctive Datalog. *The Tenth International Conference on Principles of Knowledge Representation and Reasoning KR2006* (2006).

[47] Sanderson, R. et al. 2016. *Web Annotation Data Model*.

[48] Sang, E.F. and De Meulder, F. 2003. Introduction to the CoNLL-2003 shared task: Language-independent named entity recognition. *arXiv preprint cs/0306050*. (2003).

[49] Sekine, S. et al. 2002. Extended Named Entity Hierarchy. *LREC* (2002).

[50] Sirin, E. et al. 2007. Pellet: A practical OWL-DL reasoner. *Web Semantics*. (2007). DOI:https://doi.org/10.1016/j.websem.2007.03.004.

[51] Speranza, M. 2009. The named entity recognition task at evalita 2009. *Proceedings of the Workshop Evalita* (2009).

[52] SPIN - Overview and Motivation W3C: Member Submission 22 February 2011: 2011. .

[53] Staab, S. et al. 2001. Knowledge processes and ontologies. *IEEE Intelligent Systems and Their Applications*. (2001). DOI:https://doi.org/10.1109/5254.912382.

[54] Studer, R. et al. 1998. Knowledge engineering: Principles and methods. *Data & Knowledge Engineering*. 25, 1–2 (Mar. 1998), 161–197. DOI:https://doi.org/10.1016/S0169-023X(97)00056-6.

[55] Suárez-Figueroa, M.C. et al. 2009. How to write and use the ontology requirements specification document. *Lecture Notes in Computer Science (including subseries Lecture Notes in Artificial Intelligence and Lecture Notes in Bioinformatics)* (2009).

[56] Tsarkov, D. and Horrocks, I. 2006. FaCT++ Description Logic Reasoner: System Description. Springer, Berlin, Heidelberg. 292–297.

[57] Vardi, M.Y. 1996. Why is modal logic so robustly decidable? *Descriptive Complexity and Finite Models: Proceedings of a DIMACS Workshop* (1996).

A Appendix

A.1. Software and tools

The current version of the JSON to RDF converter is available in the repository https://gitlab.com/rousi.maria1/eopen-jsontordfconverter. The converter gets as an input a JSON file, converts it into RDF, saves the RDF format in the semantic database GraphDB and returns as a result the RDF format of the input. The converter is a post web service, while a dockerfile is available in order to be integrated with docker. Also, in the repository https://gitlab.com/rousi.maria1/eopen-jsontordfclient a java client is available which sends a JSON to the converter and gets as a result the RDF format of the input.