HACID - Deliverable

Top-down and bottom-up approaches to domain knowledge engineering

This project has received funding from the European Union's Horizon Europe research and innovation programme under Grant Agreement No. 101070588. UK Research and Innovation (UKRI) funds the Nesta and Met Office contributions to the HACID project.

Deliverable number:	D2.1
Due date:	29.02.2024
Nature[1]:	R
Dissemination Level[2]:	PU
Work Package:	WP2
Lead Beneficiary:	CNR
Contributing Beneficiaries:	all

[1] The following codes are admitted:
- R: Document, report (excluding the periodic and final reports)
- DEM: Demonstrator, pilot, prototype, plan designs
- DEC: Websites, patents filing, press & media actions, videos, etc.
- DATA: Data sets, microdata, etc.
- DMP: Data management plan
- ETHICS: Deliverables related to ethics issues.
- SECURITY: Deliverables related to security issues
- OTHER: Software, technical diagram, algorithms, models, etc.

[2] The following codes are admitted:
- PU – Public, fully open, e.g. web (Deliverables flagged as public will be automatically published in CORDIS project's page)
- SEN – Sensitive, limited under the conditions of the Grant Agreement
- Classified R-UE/EU-R – EU RESTRICTED under the Commission Decision No2015/444
- Classified C-UE/EU-C – EU CONFIDENTIAL under the Commission Decision No2015/444
- Classified S-UE/EU-S – EU SECRET under the Commission Decision No2015/444

Document History

Version	Date	Description	Author	Partner
0.1	19/01/2024	Creation of draft document	Andrea Nuzzolese	CNR
0.2	14/02/2024	Description of early methodologies for top-down knowledge engineering	Anna Sofia Lippolis	CNR
		Bottom-up knowledge engineering background	Weilai Xu	
0.3	15/02/2024	Description of recent approaches for top-down knowledge engineering	Anna Sofia Lippolis	CNR
0.4	16/02/2024	The eXtreme Design methodology	Anna Sofia Lippolis and Andrea Giovanni Nuzzolese	CNR
0.5	20/02/2024	Top-level ontology and conceptual grounding	Andrea Giovanni Nuzzolese	CNR
0.6	21/02/2024	Description of core modules	Andrea Giovanni Nuzzolese	CNR
0.7	22/02/2024	Ontology for medical diagnostics	Andrea Giovanni Nuzzolese	CNR
0.8	23/02/2024	Bottom-up DKG for medical diagnostics	Weilai Xu	CNR
0.9	26/02/2024	Ontology for Climate Services	Miguel Ceriani	CNR
1.0	28/02/2024	Full text revision	Vito Trianni	CNR
1.1	05/03/2024	Corrections and improvements	Andrea Nuzzolese, Miguel Ceriani, Alessandro Russo	CNR
1.2	07/03/2024	Final Version	Vito Trianni, Andrea Nuzzolese, Miguel Ceriani	CNR

Table of content

1. Introduction

This document provides a description of the methods for the design and production of the domain knowledge graph (DKG) that is exploited in the HACID project as the backbone for the implementation of hybrid human-artificial collective intelligence. The DKG represents domain knowledge for the two case studies we have considered, namely medical diagnostics and climate change adaptation management (also referred to as climate services). In the following, we provide the background that informs the methodologies used within the project, differentiating between top-down and bottom-up knowledge engineering approaches (Section 2), we discuss the implementation choices for knowledge engineering (Section 3), and we finally present the approach chosen for information extraction with a preliminary evaluation (Section 4).

2. Background

2.1. Top-down knowledge engineering

Ontologies are not only software components, but also information artefacts whose quality is closely linked to their design, as it must accurately describe a domain and fulfil a purpose that involves identifying elements considered in the same case, and undergo a life cycle. Despite their critical role in the operation of numerous infrastructures across both industry and academia, one of the main challenges preventing the widespread adoption of ontologies is the complexity of their development. In fact, this process requires interdisciplinary expertise and practical guidance [1]. In the realm of engineering ontology applications, a predominantly top-down construction approach is embraced. This approach involves leveraging existing ontologies to derive structure and applying it to available data, typically achieved through comprehensive domain analysis and interviews with engineers [2].

As the Semantic Web gained increasing attention towards the late Nineties, researchers embarked on creating numerous Ontology Engineering Methodologies (OEMs) aimed at providing comprehensive and clear directions for ontology construction. While there is consensus among all OEMs on the essential activities required for ontology engineering—beginning with a feasibility study to assess the utility of an ontological approach for the problem at hand, followed by domain analysis to identify relevant domains and gather knowledge for formalisation, leading to the conceptualization phase where relationships among concepts are established, and culminating in the actual implementation through formal languages, resulting in a prototype ontology ready for maintenance and application—the specific approaches to these activities vary significantly.

Each OEM adopts unique approaches for specific tasks, leading to a variety of classifications based on the organisation of workflows, the degree of collaboration, and the type of support provided. Workflow strategies among OEMs vary widely, from the linear steps of the waterfall model, which enforces a strict order from a feasibility study to the release of prototypes, to lifecycle approaches that treat ontologies as continuously evolving entities without a fixed development sequence, and agile methodologies that provide ontology engineers with a flexible set of actions to tackle as necessary. The trend towards cooperative development underscores the perception of ontology engineering as a process aimed at creating a

“shared” formal representation of a domain. This highlights the essential role of domain experts and stakeholders in determining concepts and relationships for a more efficient creation of ontologies. A key finding from Kotis et al’s survey of OEMs [3] is that non-collaborative OEMs negatively affect the vitality, evolution, and reusability of ontologies. In contrast, collaborative and tailored OEMs, to positively influence the condition of developed ontologies, must be bolstered by comprehensive collaboration tools for the development, upkeep, and progression of ontologies in a modular fashion. Furthermore, OEMs vary in their emphasis, with some focusing on providing broad guidance for moving from informal domain representations to formal structures, and others on offering detailed, or authoring, methodologies support to help choose the most appropriate logical frameworks for knowledge modelling.

2.1.1 Early methodologies

METHONTOLOGY [4] emerged as one of the early structured methodologies, proposing a waterfall-like process with phases such as specification, knowledge acquisition, conceptualization, integration, implementation, evaluation, and documentation. Despite its structured approach, METHONTOLOGY introduced flexibility with overlapping phases, emphasising a comprehensive yet adaptable process for ontology development. The Ontology Development 101 guide by Noy & McGuinness (2001) [5] laid down a foundational framework pre-dating the widespread adoption of the Web Ontology Language (OWL). It described a traditional, sequential method comprising domain definition, reuse consideration, term enumeration, class hierarchy development, property definition, and instance creation. Although detailed in several steps, it lacked emphasis on requirements gathering and testing. DILIGENT [6] was instead based on theories of argumentation, focusing on empowering domain experts in ontology engineering through continuous and distributed construction and updating of ontologies. It highlighted the importance of domain expert involvement and the collaborative nature of ontology development. This same participation of knowledge workers in the ontology lifecycle was advocated by the Human-Centred Collaborative Ontology Engineering Methodology (HCOME) [7]. It emphasised a human-centred approach, supporting collaboration between domain experts and ontology engineers through direct and continuous interaction with conceptualizations.
Ontology-Grounded Methods and Applications (DOGMA) and its extension DOGMA-MESS [8] followed a data-driven, bottom-up approach to ontology engineering, enabling the modelling of shared ontologies in stakeholders’ terminology. It introduced a collaborative and iterative development process driven by social knowledge conversion modes. Melting Point [9], drawing from best practices in earlier methodologies, emphasised collaborative development by communities of practice without prescribing specific techniques, thus recommending that ontology developers consider those methods best suited to their particular situations.

2.1.2 Recent approaches

A facet-based OEM for Geospatial Ontologies [10] (FMCLGO) adopted a faceted approach for the development of large-scale geospatial ontologies. It focused on manual and automatic processes for identifying, categorising, and populating ontologies, highlighting a top-down approach to ontology engineering.

Contrary to other methodologies that offer structured guidance for ontology engineering, the NeOn Methodology [11] presents a flexible approach, proposing various paths for ontology development. Introduced in 2008, NeOn, along with Grounding Ontologies with Social Processes and Natural Language [12], enriches the ontology engineering methodology landscape. These methodologies underscore the importance of collaborative development, engaging both developers and ontology practitioners, promoting the reuse and integration of ontologies within networks, and highlighting the significance of social processes and natural language in the evolution of ontologies. NeOn specifically focuses on the dynamics of ontology networks, encouraging the collaborative creation of ontologies and the reuse and re-engineering of existing knowledge. A notable feature of the NeOn Methodology is its comprehensive library of ontology design patterns[3] [13] and the provision of tool-supported methodological guidance, known as eXtreme Design [14], to facilitate ontology development. This approach mirrors the shift in software engineering towards agile methodologies, moving away from the highly disciplined processes that dominated the field's early days towards the use of rapid and iterative processes.

Agile OEMs have emerged as a response to the demand for swift and collaborative ontology development in decentralised environments. These methodologies highlight the significance of stakeholder collaboration and the vital role of domain experts in shaping the knowledge base and underlying conceptual framework. Different studies agree on the fact that by streamlining the engineering process, agile OEMs efficiently scale down the involvement of engineers, directly addressing task-related hurdles such as knowledge retrieval, documentation creation, and conceptualization, while also shortening the learning curve attributed to their flexible structure [15]. Agile OEMs facilitate the process without necessitating stakeholders and domain experts to possess ontology modelling expertise. Instead, they recommend the use of supportive tools for identifying and conceptualising knowledge, emphasising the agile paradigm's focus on intervention areas rather than predefined sequences or steps, ultimately guiding ontology engineers toward prototype development.

According to the evaluation of Spoladore and Pessot [16], among the various agile OEMs, UPONLite [17], SAMOD[4], and RapidOWL [18] stand out as valuable resources for industry application. UPONLite, introduced in 2016, advocates for a lightweight and expedited approach to ontology engineering, emphasising a user-centred and social methodology. It promotes rapid prototyping through participatory methods, underlining the importance of domain experts and knowledge workers' involvement in the development process. The methodology is structured around six steps, from identifying domain terminology to the formalisation of the ontology, with an emphasis on collaborative work between stakeholders, domain experts, and engineers through the initial five steps, using tools like spreadsheets and conceptual maps for domain conceptualization and specification.

SAMOD, the Simplified Agile Methodology for Ontology Development introduced by Peroni in 2016, treats ontologies as evolving prototypes. It begins with a motivating scenario from which requirements are collected through competency questions, leading to the development of a model or "micro-ontology" that is iteratively refined and expanded until it meets all the ontology's requirements. This iterative process encourages the incorporation of additional complexity or scenarios to ensure the final model comprehensively answers all competency questions, symbolising the completed ontology.

---

[3] Ontology Design Patterns: http://ontologydesignpatterns.org.

[4] SAMOD: https://essepuntato.it/papers/samod-owled2016.html

RapidOWL emphasises the agile elicitation and structuring of knowledge to foster cooperation among stakeholders, ontology engineers, and domain experts. It adopts a flexible paradigm grounded in a set of values and principles, translating into ten activities that map out the ontology development process. RapidOWL's approach is distinctly non-prescriptive, granting participants the autonomy to select and sequence activities as they see fit, without any explicit dependency among them, thereby enhancing the methodology's adaptability and responsiveness to project-specific needs.
These agile OEMs collectively underscore a shift towards more dynamic, collaborative, and user-engaged processes in ontology development. By prioritising flexibility, stakeholder involvement, and the use of established design patterns and methodologies, agile OEMs aim to streamline the ontology development process, making it more accessible and responsive to the rapidly evolving requirements of diverse domains.

2.1.3 The eXtreme Design Methodology

Within the spectrum of ontology engineering methodologies, the eXtreme Design (XD) methodology stands out for its innovative approach that draws inspiration from the principles of agile software development. This methodological innovation was introduced to meet the demands for more rapid and flexible processes in ontology development, emphasising the importance of swift prototyping, iterative design, and the active involvement of stakeholders, particularly domain experts and end-users. At the heart of XD's strategy are Ontology Design Patterns (ODPs), which serve to address common and recurring challenges in design, thereby streamlining the creation and refinement of ontologies.
Before XD, pattern-based ontology engineering primarily focused on the logical level, including ontology learning and enrichment, the use of the Ontology Pre-Processor Language (OPPL), and methods for its application in logical ODP reuse, alongside the proposal of a high-level pattern language. The integration of ODPs in ontology engineering environments, such as the logical pattern templates in Protégé 3 and the template wizard for OPPL pattern definitions in Protégé 4, primarily targeted Logical ODPs. These tools also support the inclusion of Content Patterns (CPs) in ontologies through a macro-like mechanism, broadening the application scope of ODPs in ontology development.
The process kicks off with an intensive effort to gather requirements, facilitated through close collaboration between the customer and design teams. This phase often employs the use of stories, a concept borrowed from agile software development, to clearly and effectively communicate the needs and expectations for the ontology.
Following the collection of requirements, these stories are then transformed into competency questions (CQs), which act as a natural language representation of the ontological commitments that will guide the development process. These questions are important in defining the scope and functionalities that the ontology is expected to support, ensuring that the design remains aligned with the intended use cases.
A distinguishing aspect of XD is its reliance on ODPs to tackle the competency questions identified and solve common design problems, allowing developers to draw on successful patterns from past experiences rather than starting from scratch. This not only expedites the design process but also improves the quality and interoperability of the resulting ontology by integrating proven practices.
The development process under XD is iterative, with each segment of the ontology undergoing validation against the competency questions through testing. This typically involves converting CQs into SPARQL queries and running these queries on a data sample

modelled according to the ontology. Successful validation of a segment leads to its integration into the larger ontology structure, paving the way for further iterations to refine existing components or to incorporate additional CQs.
XD is characterised by several key features that contribute to its effectiveness and innovation. The methodology places a strong emphasis on the active involvement and feedback from customers or domain experts throughout the development process, ensuring that the ontology closely reflects user needs and domain-specific insights. The use of ODPs facilitates rapid prototyping, allowing for quick adjustments and iterations based on feedback and validation outcomes.

2.2. Bottom-up knowledge engineering

The objective of bottom-up knowledge engineering, colloquially referred to as such, pertains to the augmentation and refinement of knowledge graphs (KGs) through the utilization of open information extraction (OIE) methodologies. In this section, we endeavor to provide a concise exposition of extant OIE techniques, accompanied by an analysis of their respective advantages and limitations.

2.2.1. Traditional Open Information Extraction

Before the explosion of neural networks, most strategies used for OIE systems rely on the structure of the text, for example, rule-based system, learning-based system, or clause-based system. Rule-based methods use linguistic patterns or rules to identify the semantic class of a relation phrase. For instance, REVERB [19] integrates two constraints—namely, a syntactic constraint and a lexical constraint—aimed at enhancing the performance. The syntactic constraint imposes restrictions on relation phrases to adhere to specific patterns, thereby mitigating errors such as the inadvertent concatenation of distant words. Complementarily, the lexical constraint facilitates the refinement of extractions by reducing overspecified outputs. These constraints constitute integral components of the REVERB OIE system, which yields substantial improvements in both precision and recall when compared to earlier extractors such as TEXTRUNNER [20] and WOE pos [21].
Learning-based methods employ supervised or semi-supervised learning techniques to train a classifier on pre-defined relations. For instance, OLLIE [22] represents a learning-based Open Relation Extraction (ORE) system that acquires unlexicalized pattern templates from bootstrapped training data. OLLIE adeptly extracts relational tuples from text without necessitating a pre-specified vocabulary. Notably, it surpasses earlier systems by capturing relations mediated by nouns, adjectives, and various syntactic forms, while also integrating contextual cues from the sentence into its extractions.
A clause-based system for OIE is a system that uses clauses as the basic unit of information extraction. A clause is a grammatical structure that consists of a subject, a predicate, and optionally one or more complements. A clause-based OIE system identifies clauses in a natural language sentence and then extracts relations and their arguments from each clause. For example, Angeli et al. [23] propose a novel approach to produce relation triples from natural language sentences, leveraging the linguistic structure to split sentences into self-contained clauses, and then apply natural logic inference to extract the maximally specific arguments for each candidate triple. Another proposed method, called ClauseIE [24], handles complex sentences with multiple clauses, nested clauses, and non-clausal

modifiers. From the perspective of the source of the information, most of these OIE systems require shallow features, which could be lexical, syntactical, hierarchical dependencies, etc.

2.2.2. Neural KG construction method

With advancements in neural networks and pre-training methodologies, there arises an opportunity to delve into deeper semantic representations, such as embeddings, moving beyond reliance solely on surface-level features like Part-of-Speech (POS) tags. This evolution enables a more nuanced understanding and extraction of information from textual data. Overall, neural KG construction or information extraction can be defined as two problems, the discrimination (classification) problem and the generation problem. Considering the input information, these methods extensively utilise semantic information. They commonly rely on word and sentence representations derived from embedding models (e.g. ELMo) or pre-trained language models, which encapsulate both syntactic and semantic information.

2.2.2.1. Discrimination (Classification)

The discrimination methods commonly generate output tags using sequential tagging for each position. Starting from the beginning of a sentence, each position within the sentence is annotated with a tag based on a notation schema designed to identify entities. Subsequently, the methods predict relation labels between these tagged entities based on pre-defined relations. Angeli et al. [25] introduce a novel method termed bootstrapped self-training for knowledge base population, which addresses the inherent challenge of limited supervised training data in relation extraction. This approach integrates the precision of pattern-based relation extractors with the robustness of trained models. It involves iteratively refining the model through training on the outputs of patterns to enhance performance while incorporating predictions from previous iterations to improve recall. Notably, the proposed method demonstrates commendable performance on the task, even when relying on a pattern-based model. The self-trained models emerge as top-performing systems, underscoring the efficacy of the approach. However, conventional pipeline setups for slot filling systems, encompassing components such as an information retrieval system, entity linker, relation extractor, and consistency component, exhibit practical drawbacks. Notably, accuracy discrepancies between the slot filling track and the knowledge base track underscore the potential benefits of integrating consistency and inference components. Exploring constraints across the entire predicted knowledge base emerges as a promising avenue for future research.

In a separate study, Luan et al. [26] present a multi-task framework for identifying and categorizing entities, relations, and coreference clusters within scientific articles. Introducing the SCIERC dataset and the Scientific Information Extractor (SCIIE) framework, the authors develop a unified approach to address this multifaceted task. The multi-task setup mitigates cascading errors between tasks and leverages cross-sentence relations via coreference links. Notably, the multi-task model surpasses previous approaches in scientific information extraction, notably without relying on domain-specific features. Moreover, it facilitates the construction of a scientific knowledge graph, thereby enabling insightful analyses of information contained within scientific literature. While the paper does not explicitly delineate any drawbacks associated with the proposed method, further investigation may be warranted to ascertain potential limitations.

Dai et al. [27] present a unified joint extraction model that tags entity and relation labels directly based on the position of a query word, allowing for entity detection at position *p* and identification of relationships with other entities in the sentence. They devise a tagging scheme to generate *n*-tag sequences for an *n*-word sentence and introduce a position-attention mechanism to model these sequences. The method enables simultaneous extraction of all entities and their types, along with identification of overlapping relations. Experimental results demonstrate superior performance in extracting overlapping relations and detecting long-range relationships, achieving state-of-the-art results on two public datasets.

2.2.2.2. Generation

Generative methodologies that employ the sequence-to-sequence framework for constructing knowledge graphs are noted for their adaptability and versatility across diverse tasks. In essence, these approaches conceptualize the extraction process as a generation problem, wherein the sequential generation of linearized entities and relations within a triplet occurs autonomously through autoregressive mechanisms.

UIE [28] is a unified text-to-structure generation framework aimed at addressing diverse information extraction (IE) tasks while enabling the generation of targeted structures and the acquisition of general IE capabilities from multiple knowledge sources. UIE employs a structured extraction language to encode extraction structures, employs a schema-based prompt mechanism for generating target extractions, and utilizes a large-scale pre-trained text-to-structure model to encompass common IE abilities. Notably, UIE has demonstrated state-of-the-art performance across various IE tasks, datasets, and settings, including supervised, low-resource, and few-shot scenarios. Its efficacy, universality, and transferability extend to a broad spectrum of entity, relation, event, and sentiment extraction tasks, consolidating them within a unified framework. Although the paper does not explicitly delineate any disadvantages of the UIE framework, further scrutiny of the literature may be necessary to discern potential limitations or avenues for refinement.

DEEPSTRUCT [29] aims to enhance the structural comprehension capabilities of language models in structure prediction tasks. By pretraining language models on task-agnostic corpora to generate structures from text, DEEPSTRUCT facilitates the transfer of acquired knowledge to downstream tasks. Notably, DEEPSTRUCT exhibits superior performance across 21 out of 28 evaluated datasets spanning 10 structure prediction tasks, without necessitating task-specific architectures or data augmentation. In summary, DEEPSTRUCT presents a promising avenue for improving the structural comprehension abilities of language models in structure prediction tasks.

2.2.3. Retrieval-Augmented Generation

While neural-based knowledge graph construction methods have demonstrated success, they also exhibit inherent limitations. These limitations stem from the nature of the paradigm:

- Neural KG models necessitate training or fine-tuning, which is both time-consuming and costly.
- They require a substantial amount of labeled data for effective operation.
- Trained or fine-tuned neural KG models face challenges in terms of updates, e.g. adding new dataset.
- The use of pre-trained large language models (LLMs) alone may lead to hallucination problems.

Retrieval-Augmented Generation [30] (RAG) signifies a pivotal advancement in the realm of LLMs, particularly in enriching generative tasks. RAG introduces a novel paradigm where LLMs engage in an initial retrieval phase, interacting with an external data repository to gather relevant information before responding to queries or generating text. This approach not only shapes subsequent text generation but also ensures that responses are grounded in retrieved evidence, significantly enhancing the precision and relevance of the generated content. Additionally, a few strategies have been proposed to improve RAG's performance by fine-tuning embedding models [26] or fine-tuning retrievers [31].
In comparison to fine-tuning LLMs, RAG offers several distinct advantages. Firstly, RAG's retrieval-based approach allows seamless incorporation of external knowledge, providing a richer context for response generation. Unlike fine-tuning, which relies solely on training data, RAG can adapt to diverse information sources, enhancing its versatility and performance across different domains. Additionally, RAG mitigates the risk of overfitting to specific datasets by accessing fresh information during inference, reducing the likelihood of biased or outdated outputs. This adaptability and robustness make RAG a compelling choice for applications requiring accurate and contextually grounded text generation.
Regarding the input information for RAG, semantic features remain the primary component of the information source, used for embedding query prompts and retrieving the most relevant context chunks and triplets. Additionally, external expert knowledge can be incorporated either explicitly through prompt text (e.g., in-context learning strategy) or through hierarchical embedded ontologies.

3. Top-down modelling and construction of the domain knowledge graph

Methodologically, the design of the knowledge graph at the core of the HACID decision support system was based on XD, which fosters modularities and the re-use of Ontology Design Patterns (ODPs) as basic building blocks addressing common and recurring design problems. Hence, we designed the schema of our knowledge graph as a network of ontologies consisting of:

- a top-level module that provides high-level concepts;
- a set of modules that realises the core level and provides relevant domain-independent ontology patterns;
- two sets of domain ontologies, i.e. one for medical diagnostics and the other for climate services.

The general architecture of the ontology network is reported in Figure 1. In the figure, the arrows represent `owl:imports` axioms, whilst the circles represent ontology modules part of the network. The figure organises the ontology modules in three main components, i.e. (i) top-level, (ii) core modules, and (iii) domain ontologies. It is worth saying that the two domain ontologies are further divided into sub-modules, but we report them as a whole in the figure for readability.

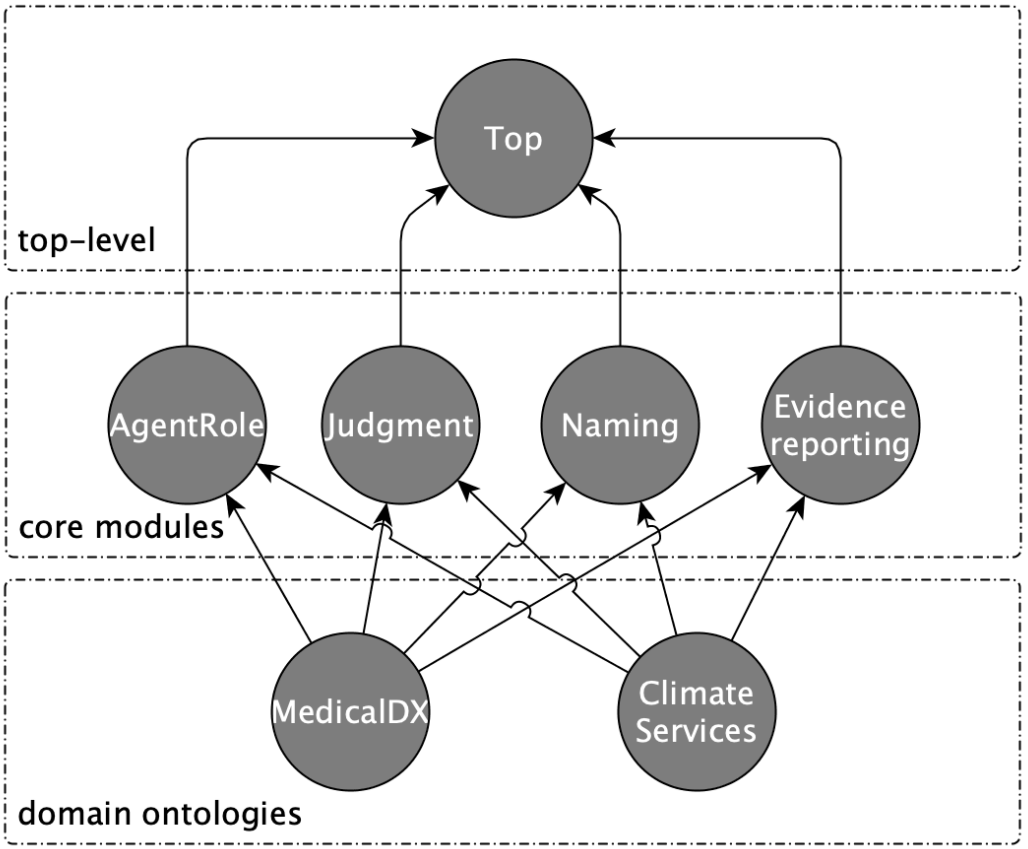

Figure 1. Architecture of the ontology network.

3.1. Top-level ontology and grounding

Figure 2 shows a fragment of the top-level module depicted with Graffoo[5] [32] as the reference notation for representing OWL graphically. The full OWL source code of such an ontology module is publicly available with a CC BY 4.0 Attribution 4.0 International[6] licence on the GitHub repository of the project[7]. We remark that all the ontologies of the network are released with the same licence.

The top-level module answers to the competency questions (CQs) reported in Table 1. We explain how the module addresses the CQs in the next paragraphs by also providing theoretical grounding with existing ontologies and patterns.

[5] https://essepuntato.it/graffoo/
[6] Licence of the ontology network: https://creativecommons.org/licenses/by/4.0/
[7] OWL source code of the top-level module: https://github.com/hacid-project/knowledge-graph/blob/main/ontologies/top-level/top-level.owl

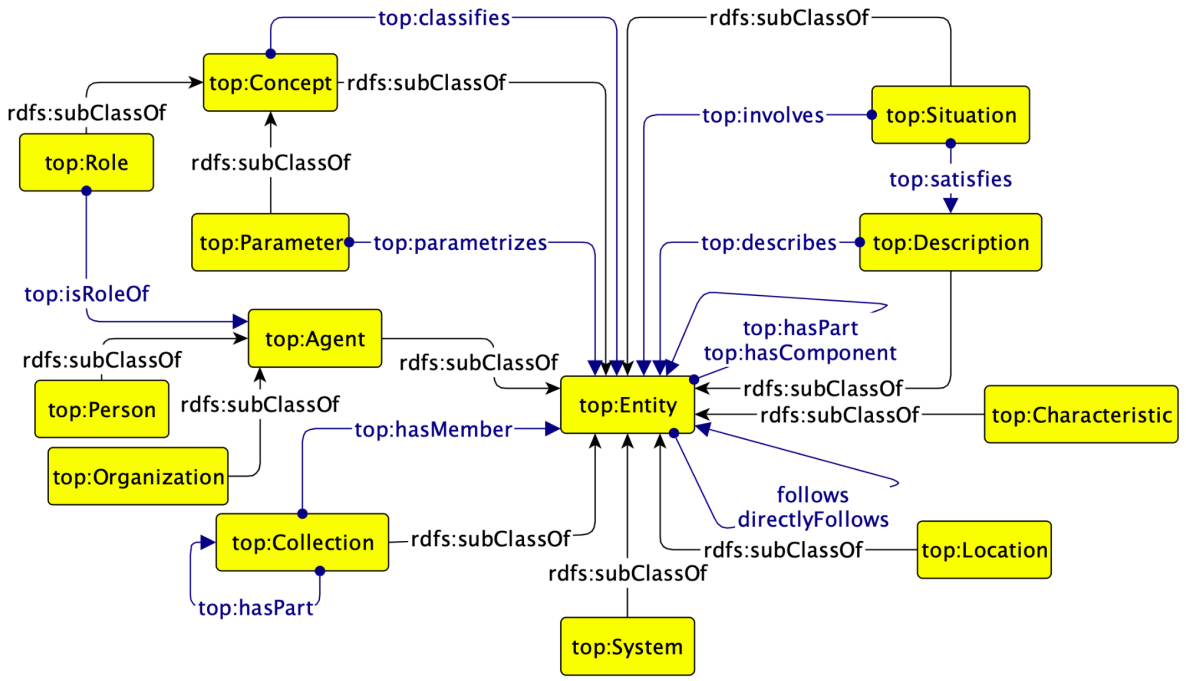

Figure 2. Top-level module, i.e. top, of the ontology network.

Table 1. Competency questions of the top-level module.

ID	Competency question
top-1	What concept is assigned to this entity?
top-2	Which category does this entity belong to?
top-3	Which agent does play a certain role?
top-4	What is the role played by a given agent?
top-5	What things are contained in a specific collection?
top-6	What collections is this thing a member of?
top-7	What is this entity part of?
top-8	What are the parts of this entity?
top-9	What is a certain object component of?
top-10	What are the components of this object?
top-11	What is before what?
top-12	What is next after a certain entity?
top-13	What is immediately following a given entity?

top-14	What is the situation that satisfies a certain description?
top-15	What are the arguments that characterise a description?
top-16	What are the entities that are involved in a situation?

We remark that the top-level module is not a foundational ontology, which is out of the scope of the project. In fact, we need an ontology network that covers a domain and is compliant with foundational ontologies for interoperability. Nevertheless, the top-level ontology is inspired by DOLCE UltraLite+DnS[8] (DUL) and DOLCE-Zero[9] (d0) in terms of top classes we defined and patterns we modelled. Both DUL and d0 are well known OWL representations of DOLCE [33], which is a foundational ontology aimed at modelling a commonsense view of reality by applying a sound approach with respect to cognitive and linguistic theories. More in detail, DUL is a commonly used foundational ontology that commits to:

- DOLCE [33] distinctions: objects vs. events vs. qualities (specific attributes of objects and events) vs. qualia (dimensional representations of qualities)
- DnS [34, 11] distinctions for second-order entities: situations vs. descriptions vs. concepts, including e.g. types, topics, roles, tasks, quality types, parameters, reified relations and classes, etc.

Following the modelling solution adopted in DUL, our top-level defines a general class named `top:Entity`[10], which is equivalent to `dul:Entity`. All the other classes are defined as subclasses of `top:Entity` and provide the core entity types that can be represented in a knowledge graph. We reuse some well-known ontology design patterns for modelling the top-level module. Namely, we reuse the following ODPs:

- the Classification pattern[11] that allows to represent the relations between concepts (e.g. roles) and any other possible entity (e.g. person), which concepts can be assigned to (i.e. `top:Concept top:classified top:Entity`). With this pattern it is possible, for example, to make assertions about categories, types, roles, which are typically considered at the top-level of an ontology. Hence the reuse of this pattern allows the top-level module to address the CQs #top-1 and #top-2 in Table 1;
- The Agent-Role pattern[12] that allows a designer to make assertions on roles played by agents without involving the agents that play those roles, and vice versa. It does not allow one to express the temporariness of roles. In our module we have the class `top:Role` that specialises `top:Concept` and is linked to `top:Agent` by means of the object property `top:isRoleOf`. The reuse of the Agent-Role pattern allows the top-level module to address the CQs #top-3 and #top-4.
- The Collection pattern[13] to represent domain membership relations. It is possible to express that a given `top:Collection` has certain individuals of `top:Entity` as members through the property `top:hasMember`, which is not transitive. The use of the collection pattern addresses the CQs #top-5 and #top-6.

[8] DUL: http://www.ontologydesignpatterns.org/ont/dul/DUL.owl
[9] d0: http://www.ontologydesignpatterns.org/ont/dul/d0.owl
[10] The prefix `top:` stands for the namespace `https://w3id.org/hacid/onto/top-level/`
[11] Classification pattern: http://ontologydesignpatterns.org/wiki/Submissions:Classification
[12] AgentRole pattern: http://ontologydesignpatterns.org/wiki/Submissions:AgentRole
[13] Collection pattern: http://ontologydesignpatterns.org/wiki/Submissions:Collection

- The PartOf[14] and Componency[15] patterns that allow one to represent entities and their parts i.e., part-whole relations, with and without transitivity, respectively. Namely, it is possible to express meronymy between a part (i.e. any `top:Entity` denoting a meronym) and its whole (i.e. any `top:Entity` denoting as a holonym) through the object property `top:isPartOf` (and its inverse property `top:hasPart`), which is formalised as transitive and reflexive. Instead, componency relations are represented by using the object property `top:hasProperPart` (and its inverse property `top:isProperOf`), which is defined as a subproperty of `top:hasPart`, but defined as asymmetric, not transitive, and not reflexive. The PartOf and Componency patterns address the CQs from #top-7 to #top-10.
- The Sequence pattern[16], which defines the notion of transitive and intransitive precedence among any possible type of entities and their inverses. It can then be used between tasks, processes, time intervals, spatially located objects, situations, etc. The pattern is applied by defining the property `top:precedes`, and its inverse `top:follows`, which are formalised as transitive. The transitive characteristic of those object properties allows an automatic reasoner to infer new sequential relations transitively. As an example, if in a knowledge graph it is stated that `:A top:precedes :B` and `:B top:precedes :C`, then `:A top:precedes :C` can be automatically inferred by means of the transitive characteristic of the object property `top:precedes`. Instead, the properties `top:directlyPrecedes` and `top:directlyFollows` allows to represent sequences without transitivity. The properties `top:directlyPrecedes` and `top:directlyFollows` properties are modelled as subproperties of `top:precedes` and `top:follows`, respectively. The adoption of the Sequence pattern allows the top-level module to address the CQs from #top-11 to #top-13.
- Descriptions and Situations (DnS), which is crucial in our top-level ontology as it highlights the separation between abstract frameworks (i.e., `top:Description`) and concrete situations (i.e., `top:Situation`). Descriptions offer the "what" (e.g. what is a disease and its characteristic relations with other entities), whilst situations present concrete instantiations of those descriptions (e.g. specific cases where a given disease is recognised). A situation always satisfies (i.e., the object property `top:satisfies`) some description. The separation between descriptions and situations provides a sound setting that can be applied to different domains of interest. As a matter of fact, we deal with two different domains of interest in HACID that provide a natural playground for experimenting with DnS modelling. DnS addresses the CQs from #top-14 to #top-16.

3.2. Core modules

Below the top-level module we defined a number of modular patterns that provide solutions to as many modelling problems that frequently occur in the process of knowledge graph construction. Those patterns identify the level of core modules in our ontology network. We refer to them as core as they provide finer grained knowledge representation schemata than the abstract one provided by top-level module, although they cannot be considered domain

[14] PartOf pattern: http://ontologydesignpatterns.org/wiki/Submissions:PartOf
[15] Componency pattern: http://ontologydesignpatterns.org/wiki/Submissions:Componency
[16] Sequence pattern: http://ontologydesignpatterns.org/wiki/Submissions:Sequence

schemata yet. These modules have been identified by interacting with domain experts from both Human DX and MET Office, analysing the input data they provided for medical diagnostics and climate services.

3.2.1. Agent role

Figure 3 shows the graphical representation of the AgentRole core module. Such a module allows to represent a specific type of situation in which an agent is associated with a role at a certain time.

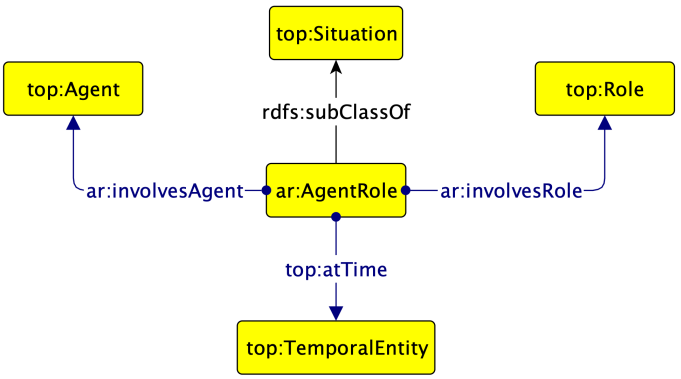

Figure 3. The AgentRole core module.

The CQs used for modelling the module are presented in Table 2.

Table 2. Competency questions of the AgentRole module.

ID	Competency question
ar-1	What is the role played by a certain agent?
ar-2	When does a given agent play a specific role?
ar-3	Who plays a specific role at a certain time?

The class `ar:AgentRole`[17] in Figure 2 implements an *n*-ary relations whose arguments are the top:Agent involved in, the `top:Role` played by such an agent, and the `top:TemporalEntity` that informs about the temporal validity of the relation between the agent and the role this agent holds. The object properties `ar:involvesAgent` and `ar:involvesRole` are both defined as sub-property of `top:involves`. The property `ar:involvesAgent` has `AgentRole` and `top:Agent` as domain and range, respectively. On the contrary, `ar:involvesRole` has `top:Agent` and `AgentRole` as domain and range, respectively.
Figure 4 shows an example of usage of the AgentRole module for representing the situation expressed by the sentence "*John was a cardiologist in December 2024.*".

[17] The prefix ar: stands for the namespace https://w3id.org/hacid/onto/core/agentrole, which identifies the AgentRole module. The OWL source code of the ontology is available on the GitHub project repository at https://github.com/hacid-project/knowledge-graph/blob/main/ontologies/core/agentrole.owl.

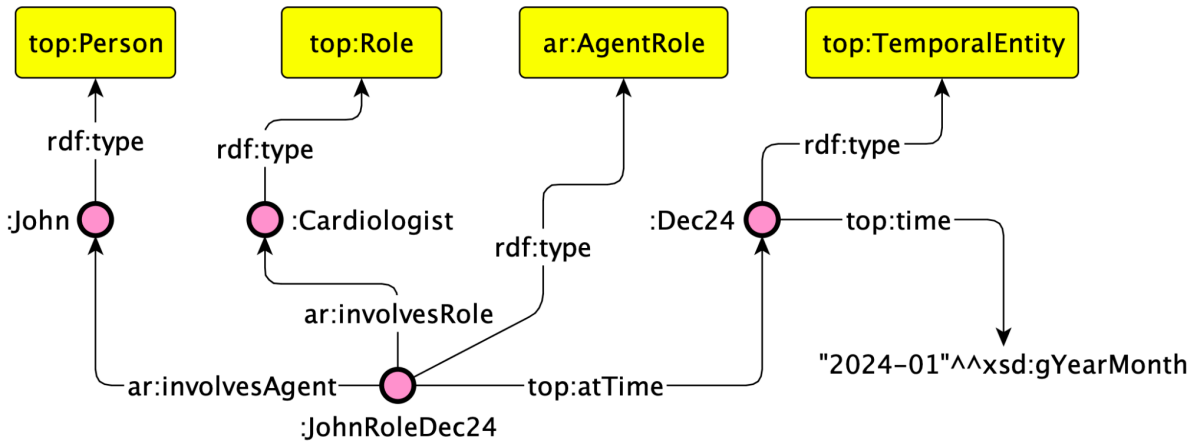

Figure 4. Example of usage of the AgentRole module.

In the example above the individual `:JohnRoleDec24` identifies the `ar:AgentRole` situation and involves the following participants:

- `:John`, which is the `top:Agent` (i.e. `top:Person` that specialises `top:Agent`) involved in the situation by means of the object property `ar:involvesAgent`;
- `:Cardiologist`, which is the top:Role involved in the situation by means of the object property `ar:involvesRole`;
- `:Dec24`, which identifies the `top:TemporalEntity` associated with the situation through the object property top:atTime. This `top:TemporalEntity` is associated with the value `"2024-01"`, which in turn is a literal typed as `xsd:gYearMonth`.

Possible SPARQL queries generated from CQs to retrieve data from a knowledge graph using the AgentRole module are the following.

- *What is the role played by a certain entity?*

```
PREFIX : <https://w3id.org/hacid/example-data/>
PREFIX ar: <https://w3id.org/hacid/onto/core/agentrole/>
SELECT ?role
WHERE {
  ?ar ar:involvesAgent :John;
      ar:involvesRole ?role
}
```

- *When does a given entity play a specific role?*

```
PREFIX : <https://w3id.org/hacid/example-data/>
PREFIX top: <https://w3id.org/hacid/onto/top-level/>
PREFIX ar: <https://w3id.org/hacid/onto/core/agentrole/>
SELECT ?role ?time
WHERE {
  ?ar ar:involvesAgent :John;
```

```
        ar:involvesRole ?role;
        top:atTime ?time
}
```

- *Who plays a role at a certain time?*

```
PREFIX : <https://w3id.org/hacid/example-data/>
PREFIX top: <https://w3id.org/hacid/onto/top-level/>
PREFIX ar: <https://w3id.org/hacid/onto/core/agentrole/>
SELECT ?agent
WHERE {
  ?ar ar:involvesAgent ?agent;
        ar:involvesRole :Cardiologist;
        top:atTime ?time .
  ?time top:time “2024-01”^^xsd:gYearMonth
}
```

3.2.2. Judgement

Judgement is a particular type of situation that in our scenarios occurs frequently. For example, a medical diagnosis or the methodology for formulating a projection for climate services are cases of judgement that require an agent to judge on a certain entity to formulate a certain decision on the entity this agent is judging on. Additionally, in our collective scenario, a specific case of judgement is identified by the assignment of a relevance score to a certain object of assessment. An example is the ranking done by a physician to the differential diagnosis she provides to a single clinical case in the context of collective medical diagnostics. Furthermore, a judgement can be the object of evaluation of another judgement that we refer to as *meta judgement*.
The CQs we identified for these scenarios are listed in Table 3, while Figure 5 shows the Graffoo diagram representing the Judgment core module.

Table 3. Competency questions of the Judgement module.

ID	Competency question
jdg-1	What is the object of a judgement?
jdg-2	Who is judging a certain entity?
jdg-3	What is the result of a judgement?
jdg-4	What is the judgement object of another judgement?
jdg-5	What is the relevance associated with an object by a certain judge?

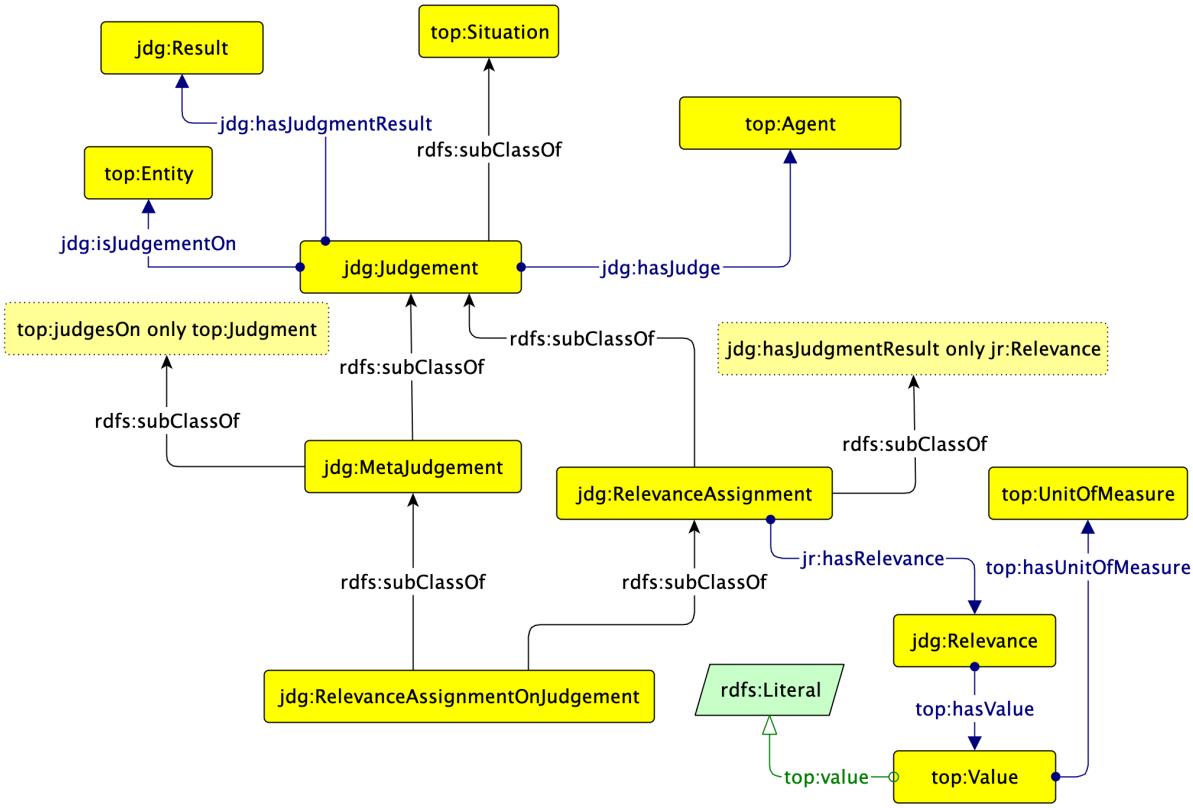

Figure 5. The Judgement core module.

In this module the class `jdg:Judgement`[18] represents a judgement situation that involves the following entities:

- the object of the judgement, which is an instance of `top:Entity` linked to the `jdg:Judgement` by means of the object property `jdg:isJudgementOn`. This allows to judge on any possible entity that exists in a KG modelled with this module;
- the judge, which is an instance of `top:Agent` linked to the `jdg:Judgement` by means of the object property `jdg:hasJudge`;
- the result of a judgement, which is an instance of `jdg:Result` linked to the `jdg:Judgement` by means of the object property `jdg:hasJudgementResult`.

Then, a `jdg:Judgement` is further specialised by the class `jdg:MetaJudgement` that represents the situation having a judgement as the object of evaluation. Hence, for a `jdg:MetaJudgement` the only possible values for the object property `jdg:judgesOn` are individuals of `jdg:Judgement`.

Instead, the class `jdg:RelevanceAssignment` specialises `jdg:Judgement` in case the result of a judgement can be an instance of `jdg:Relevance` only. An instance of `jdg:Relevance` provides the relevance that can be associated with an entity as the output of a judgement. For example, it can be an ordinal value (e.g. 1, 2, 3, etc.) in case of rankings.

[18] The prefix `jdg:` stands for the namespace `https://w3id.org/hacid/onto/core/judgement/`, which identifies the Judgement module. The OWL source code of the ontology is available on the GitHub project repository at https://github.com/hacid-project/knowledge-graph/blob/main/ontologies/core/judgement.owl.

Finally, `jdg:RelevanceAssignmentOnJudgement` combines the behaviour of both `jdg:RelevanceAssignment` and `jdg:MetaJudgement` by allowing one to have a ranking as the result of a meta judgement.
Figure 6 shows a possible instantiation of the Judgement module for the scenario identified by the sentence: *"Robert thinks that the given clinical case is a case of pneumonia".*

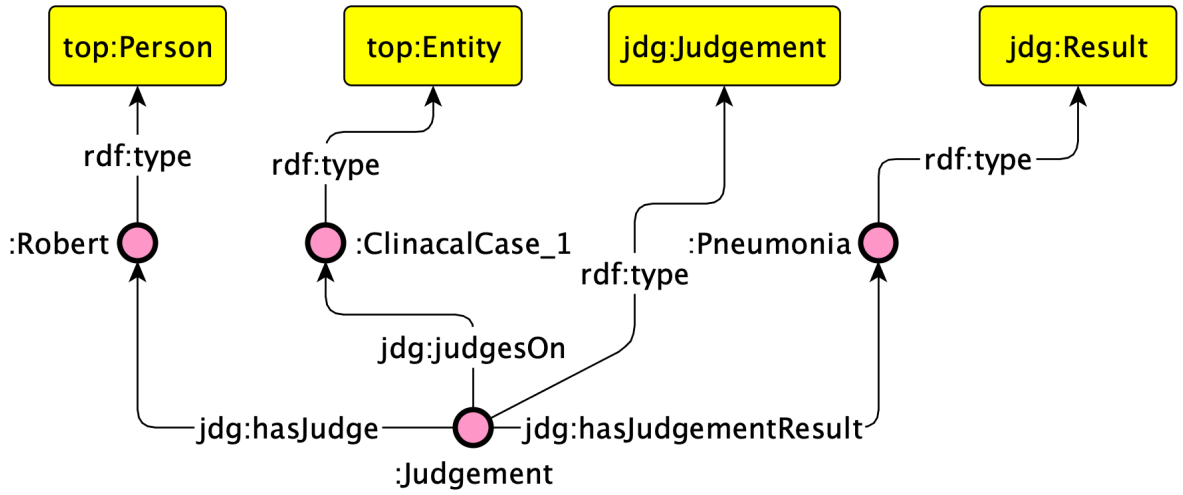

Figure 6. Example of judgement.

Possible SPARQL queries generated from CQs to retrieve data from the example reported in Figure 6 are listed as follows.

- *What is the object of a judgement?*

```
PREFIX : <https://w3id.org/hacid/example-data/>
PREFIX jdg: <https://w3id.org/hacid/onto/core/judgement/>
SELECT ?obj
WHERE {
  :Judgement jdg:judgesOn ?obj
}
```

- *Who is judging a certain entity?*

```
PREFIX : <https://w3id.org/hacid/example-data/>
PREFIX jdg: <https://w3id.org/hacid/onto/core/judgement/>
SELECT ?agent
WHERE {
  :Judgement jdg:hasJudges ?agent
}
```

- *What is the result of a judgement?*

```
PREFIX : <https://w3id.org/hacid/example-data/>
```

```
PREFIX jdg: <https://w3id.org/hacid/onto/core/judgement/>
SELECT ?result
WHERE {
  :Judgement jdg:hasJudgementResult ?result
}
```

Figure 7 shows, instead, a possible instantiation of the Judgement module for a scenario involving a meta judgement like the one in the following sentence: "*Anthony thinks that Robert's judgement is good*".

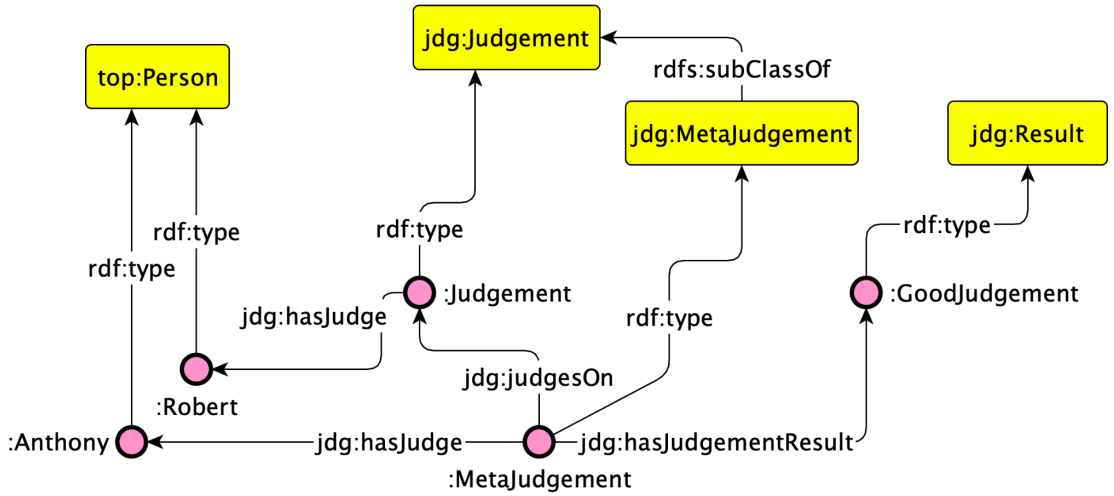

Figure 7. Example of meta judgement.

A Possible SPARQL query for the case reported in Figure 7 answers to CQ #23 as in the following listing.

- *What is the judgement object of another judgement?*

```
PREFIX : <https://w3id.org/hacid/example-data/>
PREFIX jdg: <https://w3id.org/hacid/onto/core/judgement/>
SELECT ?judgement ?result ?metaResult
WHERE {
  :MetaJudgement a jdg:MetaJudgement;
                 jdg:judgesOn ?judgement;
                 jdg:judgementResult ?metaResult .
  ?judgement a jdg:Judgement;
             jdg:judgementResult ?result .
}
```

Similarly, Figure 8 shows a possible instantiation of the Judgement module a the scenario involving a relevance assignment like the following: "*Martina ranks the solution #1 with 4 points out of 5*".

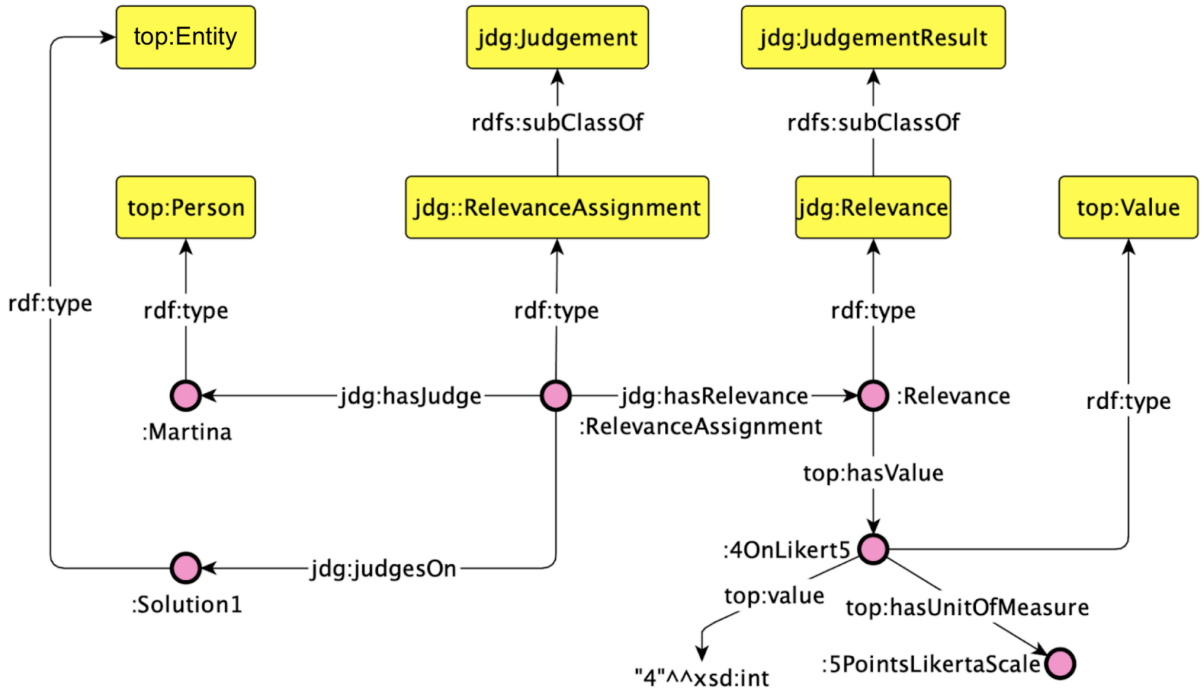

Figure 8. Example of relevance assignment.

A Possible SPARQL query for the case reported in Figure 8 answers to CQ #24 as in the following listing.

- *What is the relevance associated with an object by a certain judge?*

```
PREFIX : <https://w3id.org/hacid/example-data/>
PREFIX top: <https://w3id.org/hacid/onto/top-level/>
PREFIX jdg: <https://w3id.org/hacid/onto/core/judgement/>
SELECT ?object ?value ?unit
WHERE {
  ?relAssign a jdg:RelevanceAssignment;
               jdg:judgesOn ?object;
               jdg:hasJudge :Martina;
               jdg:hasRelevance ?relevance .
  ?relevance a jdg:Relevance;
             top:value ?value;
             top:hasUnitOfMeasure ?unit .
}
```

3.2.3. Evidence reporting

In many situations we would like to associate an information object (e.g. an article, book, webpage, etc.) to an element of the knowledge graph. Hence, the information object is an evidence available in literature about the knowledge formalised in the knowledge graph. The kind of modelling requirement is not exactly related to provenance. In the case of

provenance, we would be much more interested in gathering procedural knowledge to represent how an entity of a KG is derived from a source. However, we are interested in identifying any information object that supports the knowledge formalised within the graph, i.e., we have evidence in literature about possible declarative or factual knowledge we represent in a KG. This means that the knowledge within the KG is not derived from the information object, but it is a reference of such an information object. Provenance, instead, can be represented by relying on workflows or existing ontologies such as PROV-O[19].

The CQ we identified are reported in Table 4.

Table 4. Competency questions of the Evidence reporting module.

ID	Competency question
ev-1	What is the information entity about a certain entity in the KG?
ev-2	Which triples in the KG are a reference of a given information entity?
ev-3	Which graphs are a reference of a given information entity?

Figure 9 shows the diagram for the Evidence reporting module.

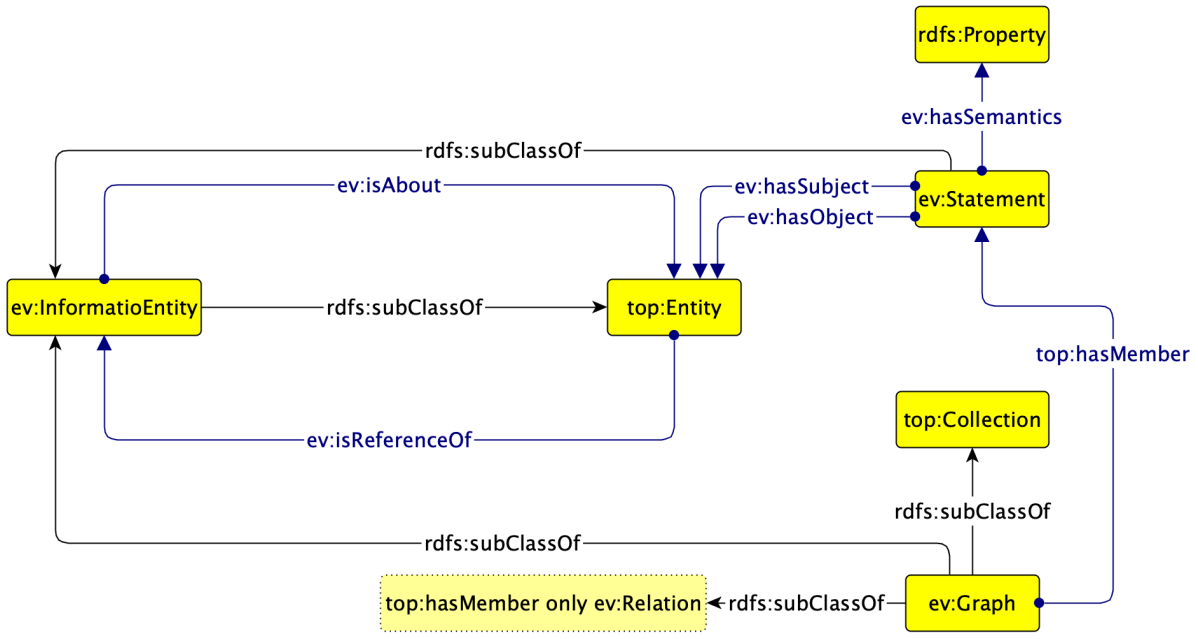

Figure 9. The Evidence reporting module.

A `ev:InformationEntity`[20] is a piece of information, such as a text, an article, a book, etc. independently from how it is concretely realised. It can be linked to any `top:Entity` of a KG by means of the object property `ev:isAbout` or, viceversa, its inverse property `ev:isRefefenceOf`. Additionally, this module specialises a `top:Entity` in order to

[19] PROV-O: https://www.w3.org/TR/2013/REC-prov-o-20130430/

[20] The prefix jdg: stands for the namespace https://w3id.org/hacid/onto/core/evidence/, which identifies the Evidence reporting module. The OWL source code of the ontology is available on the GitHub project repository at https://github.com/hacid-project/knowledge-graph/blob/main/ontologies/core/evidence.owl.

associate a `ev:InformationEntity` with a statement, i.e. `ev:Statement`, or a collection of statements, i.e. a `ev:Graph`. The class `ev:Statement` replicates in this OWL ontology the notion of reification as it is formalised in RDF. For example, this allows us to associate a document about the method applied to predict extreme events with predictions generated with such a method and published in a KG.

Figure 10 shows a usage example of the Evidence reporting module. In such an example, the entity `:10.5194/npg-18-295-2011` identifies the scientific article titled "Extreme events: dynamics, statistics and prediction" by Ghil et al. This article is linked by the `:thames-graph`, which is an individual of `ev:Graph`, through the property `ev:isReferenceOf`. This graph has the entity `:tames-stmt`, which is an individual of `ev:Statement` for representing the reification of the triple `:river-thames :floodPredictionMethod :singular-spectrum-analysis` that states that the singular spectrum analysis is used as flood precision method for the river Thames. The dotted arrow in Figure 10 identifies the relations that can be inferred through of the *property chains* defined in the module for `ev:isReferenceOf`, namely:

- `ev:hasSubject o top:isMemberOf o ev:isReferenceOf`
  `→ ev:isReferenceOf`
- `ev:hasObject o top:isMemberOf o ev:isReferenceOf`
  `→ ev:isReferenceOf`
- `top:isMemberOf o ev:isReferenceOf`
  `→ ev:isReferenceOf`

A property chain is a logical operation that allows one to derive a new object property by composing two or more object properties in a chain. In relational database work, this corresponds to performing an *equijoin* between two elementary relations and then projecting on the first and last attribute in the chain. In OWL, the equijoin operation is known as a composition operation, and an expression that composes two or more predicates is called a property chain. In the expression above the property chain is expressed syntactically by the symbol `o`. For example, the following property chain

`top:isMemberOf o ev:isReferenceOf → ev:isReferenceOf`

means that, if it is asserted that `x op:isMemberOf y` and `y ev:isReferenceOf z`, then `x ev:isReferenceOf z` is inferred.

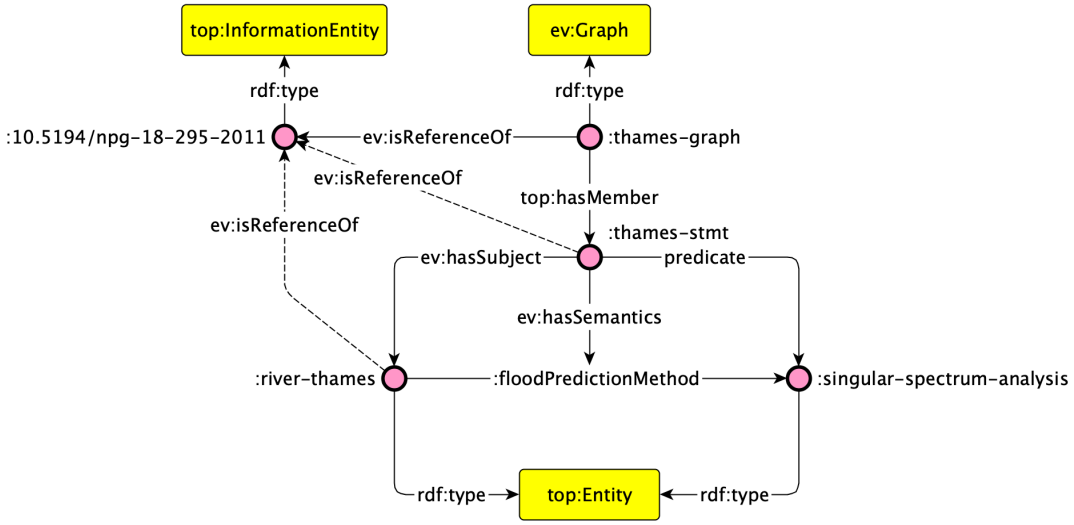

Figure 10. Usage example of the Evidence reporting module.

The SPARQL in the snippet below can be generated from CQ #ev1.

- *Which graphs are a reference of a given information entity?*

```
PREFIX : <https://w3id.org/hacid/example-data/>
PREFIX top: <https://w3id.org/hacid/onto/top-level/>
PREFIX ev: <https://w3id.org/hacid/onto/core/evidence/>
SELECT ?e ?ie
WHERE {
  ?e ev:isReferenceOf ?ie .
  ?ie a top:InformationEntity
}
```

Instead, next SPARQL query covers CQ #ev2.

- *Which triples in the KG are a reference of a given information entity?*

```
PREFIX : <https://w3id.org/hacid/example-data/>
PREFIX top: <https://w3id.org/hacid/onto/top-level/>
PREFIX ev: <https://w3id.org/hacid/onto/core/evidence/>
SELECT ?g ?e
WHERE {
  ?stmt a ev:Statement;
     ev:isReferenceOf ?ie .
  ?ie a top:InformationEntity
}
```

Similarly, next SPARQL query covers CQ #ev3.

- *Which graphs are a reference of a given information entity?*

```
PREFIX : <https://w3id.org/hacid/example-data/>
PREFIX top: <https://w3id.org/hacid/onto/top-level/>
PREFIX ev: <https://w3id.org/hacid/onto/core/evidence/>
SELECT ?g ?ie
WHERE {
  ?g a ev:Graph ;
     ev:isReferenceOf ?ie .
  ?ie a top:InformationEntity
}
```

3.2.4. Naming

Naming is a situation that involves the assignment of a name to an entity. However, an entity might be referred to with different names. Among those names, one can be preferred and others alternative. The Naming module aims at reflecting this scenario. Namely, the following are the CQs we used for modelling this module.

Table 5. Competency questions of the Naming module.

ID	Competency question
nm-1	What is the preferred name of an entity?
nm-2	What is the alternative name of an entity?
nm-3	What are the synonyms of a given label?

Figure 11 shows the Graffoo diagram for the Naming module.

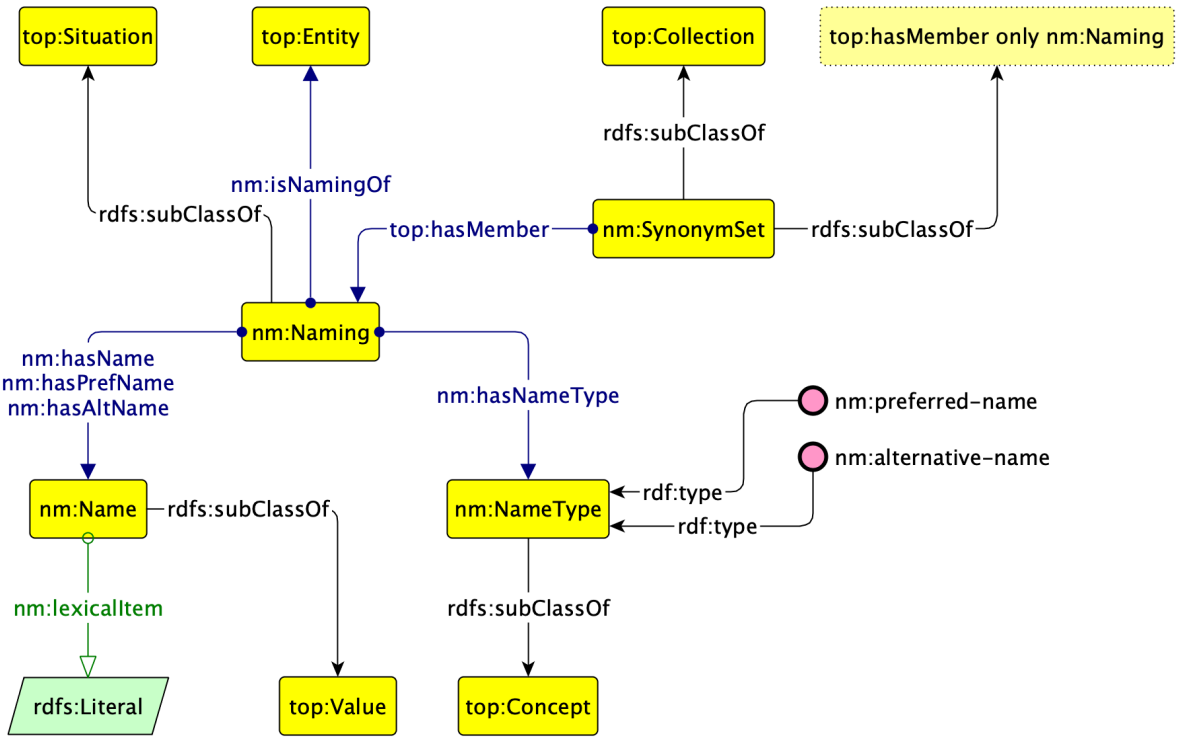

Figure 11. The Naming core module.

In Figure 11 the class `nn:Naming`[21] represents a situation in which a name is associated with a certain `top:Entity` of interest. This association is materialised through the object

[21] The prefix nm: stands for the namespace https://w3id.org/hacid/onto/core/naming/, which identifies the Naming module. The OWL source code of the ontology is available on the GitHub project repository at https://github.com/hacid-project/knowledge-graph/blob/main/ontologies/core/naming.owl.

property `nm:isNamingOf` having `nn:Naming` and `top:Entity` as domain and range, respectively. Then a `nn:Naming` is linked to an object that identifies the lexical representation of the name, i.e. `nn:Name`. The latter is associated with the literal value consisting of the lexical representation of a name through the datatype property `nm:lexicalItem`. The class `nm:NameType` allows us to classify a `nn:Naming` with respect to given categories, such as `nn:preferred-name` or `nm:alternative-name` for distinguishing between preferred names and alternative ones, respectively. This allows to tailor naming categories that reflect the needs of a domain at the extensional layer of the knowledge graph without modifying the intensional layer consisting of the ontology network. Finally, a `nm:SynonymSet` is the collection of different individuals of `nn:Naming` that are considered as synonyms.

Figure 12 shows a usage example for the Naming module that addresses the scenario covered by the following sentence: *"'COVID' and 'Disease caused by severe acute respiratory syndrome coronavirus 2' are synonyms for referring to the same disorder, i.e. COVID-19. Nevertheless, COVID should be preferred between the two."*

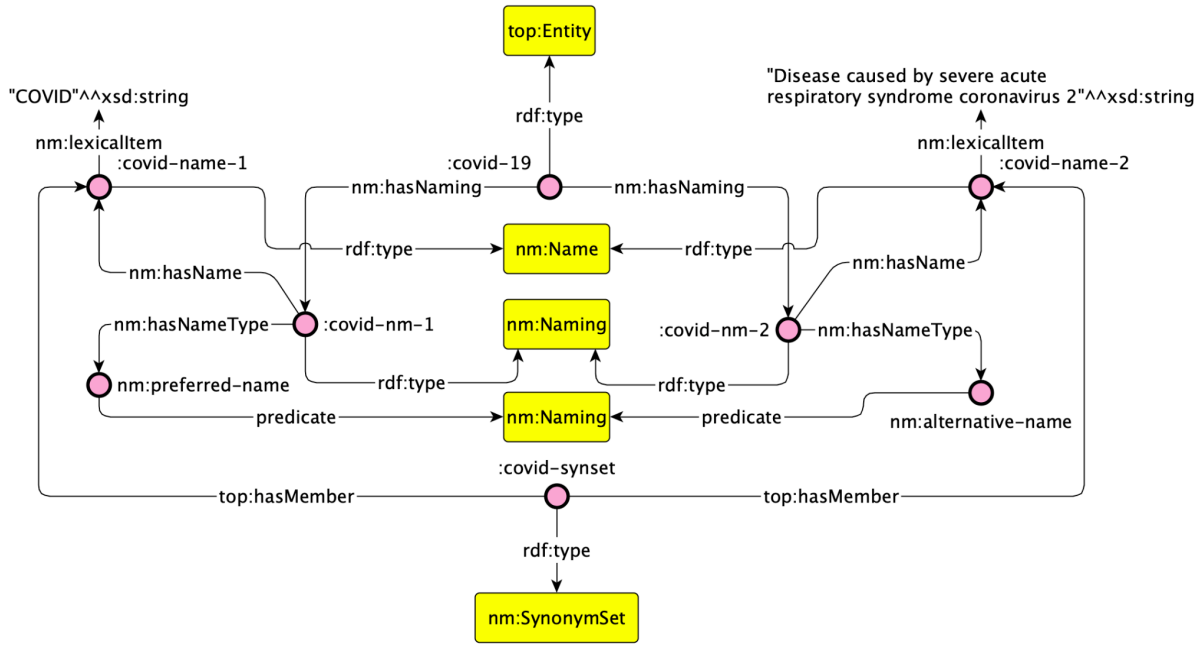

Figure 12. Example of usage of the Naming module.

According to the scenario there are:

- an entity for representing COVID-19, i.e. `:covid-19`, that can be associated with two different namings that are `:covid-nm-1` and `:covid-nm-2`, respectively;
- the naming `:covid-nm-1`, which is classified as preferred, i.e. `nm:hasNameType nm:preferred-name`, and links the `nm:Name` identified by `:covid-name-1`, which in turn links the literal `"COVID"^^xsd:string` by means of the predicate `nm:lexicalItem`;
- the naming `:covid-nm-2`, which is classified as alternative, i.e. `nm:hasNameType nm:alternative-name`, and links the `nm:Name` identified by `:covid-name-2`, which in turn links the literal `"Disease caused by severe acute respiratory syndrome coronavirus 2"^^xsd:string` by means of the predicate `nm:lexicalItem`;

- a `nm:SynonymSet` that aggregates both namings.

It is worth noticing that entities associated with namings represent real world objects, whilst synonym sets represent sets of synonyms for referring to those objects lexically. The distinction between objects of the world and synonyms is crucial when, for instance, two entities exist in two subgraphs of the same KG, but they are equivalent, allowing a single synonym set for referring to them. Similarly, the same entity might be referred to different synonym sets in different contexts, e.g. different domains such as medical diagnostics or climate services.

A Possible SPARQL query for the example reported in Figure 10 answers to CQ #nm1 as in the following listing.

- *What is the preferred name of an entity?*

```
PREFIX : <https://w3id.org/hacid/example-data/>
PREFIX top: <https://w3id.org/hacid/onto/top-level/>
PREFIX nm: <https://w3id.org/hacid/onto/core/judgement/>
SELECT ?lexicalItem
WHERE {
  :covid-19 a top:Entity ;
            nm:hasNaming ?naming .
  ?naming nm:hasName/nm:lexicalItem ?lexicalItem ;
          nm:hasNameType nm:preferred-name
}
```

Similarly, a possible SPARQL query for the example reported in Figure 10 answers to CQ #nm2 as in the following listing.

- *What is the* alternative *name of an entity?*

```
PREFIX : <https://w3id.org/hacid/example-data/>
PREFIX top: <https://w3id.org/hacid/onto/top-level/>
PREFIX nm: <https://w3id.org/hacid/onto/core/judgement/>
SELECT ?lexicalItem
WHERE {
  :covid-19 a top:Entity ;
            nm:hasNaming ?naming .
  ?naming nm:hasName/nm:lexicalItem ?lexicalItem ;
          nm:hasNameType nm:alternative-name
}
```

Finally, a possible SPARQL query for the example reported in Figure 10 answers to CQ #nm3 as in the following listing.

- *What are the synonyms of a given label?*

```
PREFIX : <https://w3id.org/hacid/example-data/>
PREFIX top: <https://w3id.org/hacid/onto/top-level/>
PREFIX nm: <https://w3id.org/hacid/onto/core/judgement/>
SELECT ?synset ?name2
WHERE {
  ?name a nm:Name ;
        nm:lexicalItem "COVID" .
  ?synset top:hasMember ?name .
          top:hasMember ?name2
  FILTER(?name != ?name2)
```

3.3. DKG for medical diagnostics

The domain ontology for medical diagnostics was generated by:

- analysing the JSON dataset provided by Human DX with clinical cases and associated *solutions* (i.e. diagnoses) submitted by *solvers* (i.e. physicians using the Human DX app);
- reframing SNOMED-CT International Edition at its version as on 30th July 2023 to get a DnS compliant representation of the domain conceptual modelling.

3.3.1. Core module from data analysis

The analysis of the dataset of Human DX was performed with the help of Human DX that was involved in a number of interactions for understanding domain and the terminology. For simplicity, the dataset can be summarised by a scenario like the following:

"A case of internal medicine reports about a patient that is found with painless jaundice, pruritus perceived as debilitating, weight loss of 15 pounds over 3 months with associated fatigue and anorexia. Additionally, the patient is affected by type 2 diabetes mellitus with difficulty controlling insulin, and hyperbilirubinemia.
A single case can be associated with a list of solutions provided by the creator of the case. If more than one solution is provided by the creator, then one must be identified as primary and other as secondary.
The case is evaluated by 5 healthcare professionals who formulated their diagnoses by taking into account the findings. Each healthcare professional provides a collection of ordered diagnoses. The ordering is based on relevance.
Additionally, each healthcare professional is associated with personal details about specialty, seniority, country, and organisation of affiliation."

Based on the previous scenario we identified the CQs as reported in Table 6. Starting from the CQs reported in Table 6 we modelled the main domain module of the ontology for the medical diagnostics. This module is shown in Figure 13.

Table 6. Competency questions of the medical diagnostics.

ID	Competency question
mdx-1	What are the specialties of a clinical case?
mdx-2	What are the diagnoses suggested by healthcare professionals?
mdx-3	Which is the relevance order of a diagnosis provided by a certain healthcare professional?
mdx-4	What is the seniority, specialty, and organisation of a healthcare professional that performs a diagnosis?
mdx-5	What is the disorder a diagnosis targets?
mdx-6	What are the findings reported for a case?
mdx-7	What are the solutions of a case?

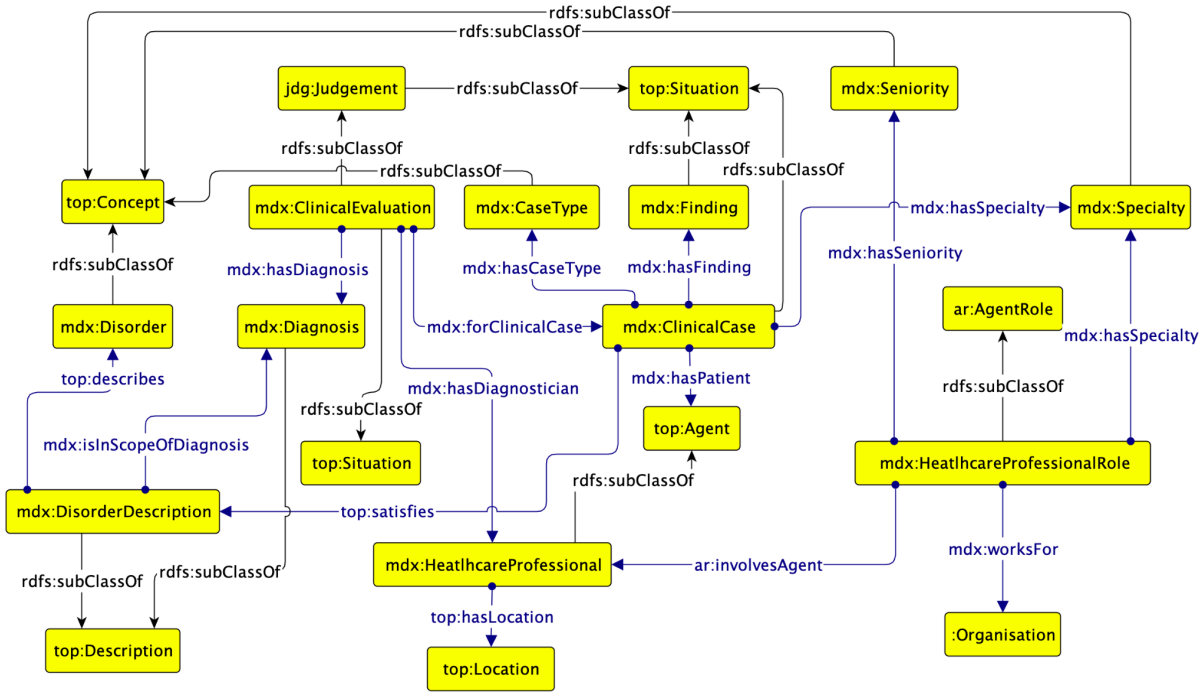

Figure 13. The main domain module of the ontology for the medical diagnostics.

A case is represented by the class `mdx:ClinicalCase`[22] that specialises `top:Situation` defined in the top-level module. Hence, the arguments of `mdx:ClinicalCase` situation are:

- a patient, which is a `top:Agent` linked to the case by the property `mdx:hasPatient`. The latter is a sub-property of `top:involves` having `mdx:ClinicalCase` and `top:Agent` as domain and range, respectively;

[22] The prefix mdx: stands for the namespace https://w3id.org/hacid/onto/mdx/, which identifies the Evidence reporting module. The OWL source code of the ontology is available on the GitHub project repository at https://github.com/hacid-project/knowledge-graph/blob/main/ontologies/medical-dx/mdx.owl.

- some findings as instances of `mdx:Finding`, which specialises `top:Situation`;

Then, the case is classified by a `mdx:Specialty` (e.g. “internal medicine”) and a `mdx:CaseType`, which is a category used for distinguishing solved cases from unsolved ones.

A `mdx:ClinicalEvaluation` is made by a `mdx:HeatlhcareProfessional` for a `mdx:ClinicalCase` and produces a `mdx:Diagnosis` as result. The class `mdx:ClinicalEvaluation` specialises `jdg:Judgement` from the Judgement core module. Hence, the following characterisation is applied:

- `mdx:forClinicalCase` specialises `jdg:judgesOn` with `mdx:ClinicalEvaluation` as domain and `mdx:ClinicalCase` as range;
- `mdx:hasDiagnostician` specialises `jdg:hasJudges` with `mdx:ClinicalEvaluation` as domain and `mdx:HeatlhcareProfessional` as range. The latter is a subclass of `top:Agent`;
- `jdg:hasResult` can have only `mdx:Diagnosis` as possible values in the context of a `mdx:ClinicalEvaluation`. This is done by adding a restriction based on universal quantification on the range, i.e. `mdx:ClinicalEvaluation` $\sqsubseteq \forall$ `jdg:hasResult.mdx:Diagnosis`.

A `mdx:Diagnosis` is a subclass of `top:Situation` that satisfies instances of `mdx:DisorderDescription` only, which in turn specialises `top:Description` and `top:describes` an instance of `mdx:Disorder` only.

Finally, the details associated with a `mdx:HealthcareProfessional` are represented by specialising the `ar:AgentRole` class from its corresponding module.

The ranking of `mdx:ClinicalEvaluation` is enabled by a specialisation of `jdg:RelevanceAssignmentOnJudgement`, i.e. `mdx:RankedClinicalEvaluation`, as presented in Figure 14. A `mdx:RankedClinicalEvaluation` associates an `mdx:Rank`, which is a `top:Value` for specialising `jdg:Relevance`, for ranking an `mdx:ClinicalEvaluation`. A `mdx:RankedClinicalEvaluation`, can be a member of a specific type of `top:Collection`, that is the `mdx:RankedClinicalEvaluationCollection`, whose objective is to aggregate instances of `mdx:RankedClinicalEvaluation` relatively to a single `mdx:ClinicalCase`.

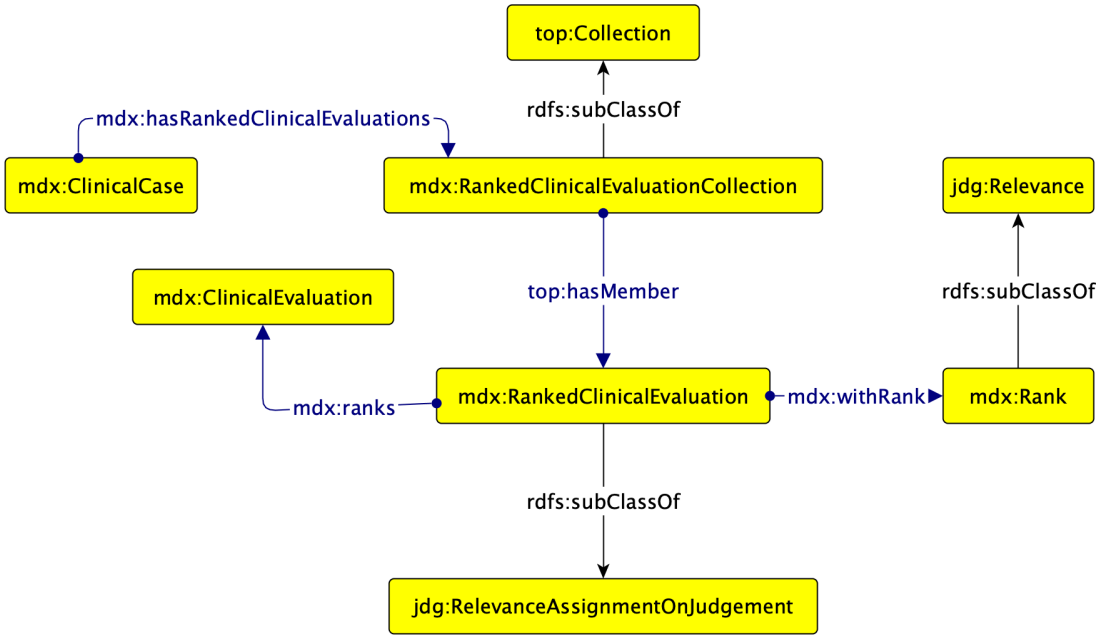

Figure 14. Modelling of ranked clinical evaluations.

3.3.2. Domain knowledge from SNOMED-CT

SNOMED-CT (Standardised Language for Healthcare, also known as SNOMED Clinical Terms, is a powerful tool in the healthcare world. It provides a structured clinical vocabulary, including a comprehensive set of diseases, morphological abnormalities, organisms and more. SNOMED-CT has a complex internal structure as it is organised as a poly-hierarchy, i.e. a hierarchy where some concepts may have more than one parent concept. Nevertheless, this poly-hierarchy is designed coherently around a set of general concepts that basically identifies the top-level of the hierarchy from the conceptual perspective. SNOMED-CT refers to those concepts as domain concepts, which are:

- **Body Structure:** it includes anatomical structures and morphologic abnormalities;
- **Clinical Finding and Disorder:** it includes observations such as the active acquisition of subjective or objective information from a primary source (clinical findings) and abnormal clinical states (disorders);
- **Environment and Geographical Location:** it includes types of environments and named locations such as countries, states, or regions;
- **Event:** it includes occurrences of events impacting health or health care, but not procedures or interventions;
- **Observable Entity:** it includes all possible entities that can be observed clinically. An observable entity is important for representing questions or assessments which can produce an answer or result;
- **Organism:** it includes any possible organism, which is significant to human medicine;
- **Pharmaceutical and Biologic Product:** it provides a top-level hierarchy to clearly distinguish drug products (products) from their chemical constituents (substances);

- **Physical Force:** it includes any concept that represents the application of energy or effort to exert pressure, impact, or influence on an object or substance, resulting in a change in its state of motion, deformation, or overall condition;
- **Physical Object:** it identifies any physical devices relevant to healthcare or to injuries/accidents;
- **Procedure:** it includes concepts representing activities performed in the provision of health care;
- **Qualifier Value:** it includes a wide range of concepts that provide attribute values used in the definitions of other concepts. These values can also be used in expressions to refine the meaning of a concept, or in the appropriate fields of a health record to add additional information;
- **Record Artefact:** it is any possible clinical documents, or parts thereof;
- **Situation with Explicit Context:** it includes concepts having *context* information; a subtype of the situation to which it applies, with an attribute associating it with the relevant clinical finding or procedure;
- **Social Context:** it represents social aspects affecting patient health and treatment. Conditions and circumstances related to healthcare that are subtypes of this hierarchy include: (i) ethnic group; (ii) lifestyle; (iii) occupation; (iv) person; (v) racial group; (vi) religion/philosophy; and (vii) social status;
- **Specimen:** it includes entities that are obtained (usually from patients) for examination or analysis;
- **Staging and Scales:** it includes concepts which are named, authoritative, and internationally relevant staging or grading systems used to either make a judgement about the patient, e.g. cognition, or, evaluate a patient to determine the phase, or progression of a disease.
- **Substance:** it contains concepts that can be used for recording and modelling: chemical constituents of medicinal and non-medicinal products; allergies, adverse reactions, poisoning; physicians and nursing orders and laboratory reports and results.

We do not take into account two other top-level domain concepts defined in SNOMED-CT that are (a) SNOMED CT Model Component and (b) Special Concept. The reasons are: (a) provides concepts and attributes necessary to organise and structure SNOMED CT terminology, whilst (b) organises concepts that are no longer active in the terminology.

Each domain concept of SNOMED-CT was used for generating a domain ontology pattern following the modelling solution provided by the DnS pattern. Each pattern we derived is available on GitHub[23]. This means that each domain concept specialises the class `top:Description` defined in the top-level module. Accordingly, all possible narrower terms available in the hierarchies of the domain concepts are then transformed to individuals of those concepts represented as description classes. Figure 15 shows the representation of this modelling solution for an excerpt of the Disorder domain concept that is presented as a subclass of `top:Description`, i.e. `mdx:DisorderDescription`[24]. The description `mdx:DisorderDescription` is associated with the original concept, i.e. `mdx:Disorder`, through the property `top:describes` inherited from the top-level module.

[23] Patterns generated from SNOMED-CT: https://github.com/hacid-project/knowledge-graph/tree/main/ontologies/medical-dx/pattern

[24] The prefix `mdx:` stands for the namespace `https://w3id.org/hacid/onto/mdx/`.

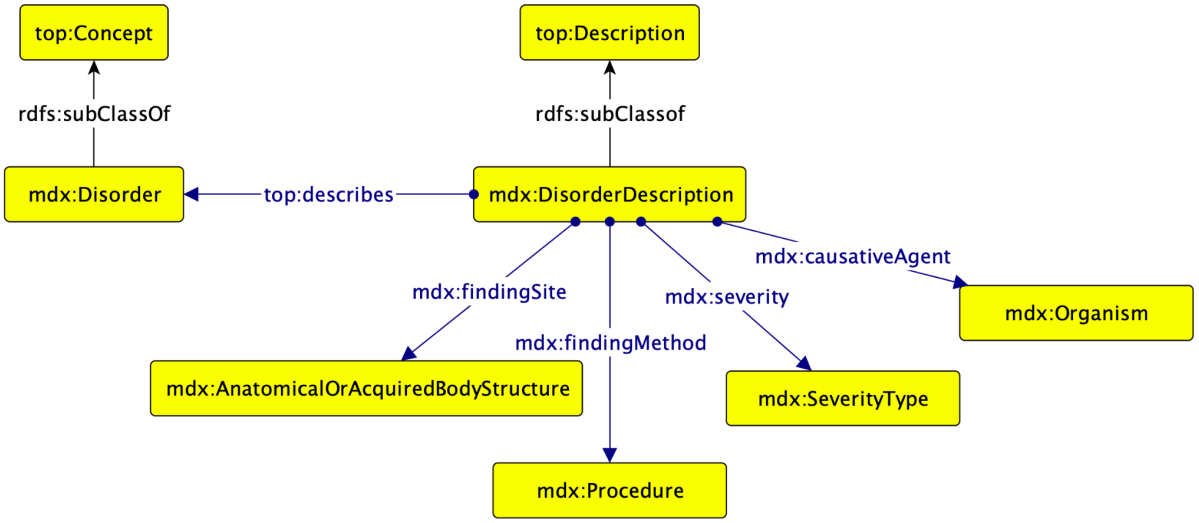

Figure 15. The Disorder domain concept as Description.

In the figure `mdx:DisorderDescription` is presented with only a few of its arguments. The full representation of the pattern can be obtained on GitHub[25].

A concept in SNOMED-CT can be associated with multiple relationship groups. A relationship group puts together attribute-value pairs to refine the meaning of a concept. When a concept has multiple relationship groups, this means that it has multiple descriptions. For example, the concept "*Electrical burn of skin (disorder)*"[26], is modelled in SNOMED-CT with two relationship groups, as shown in Figure 16.

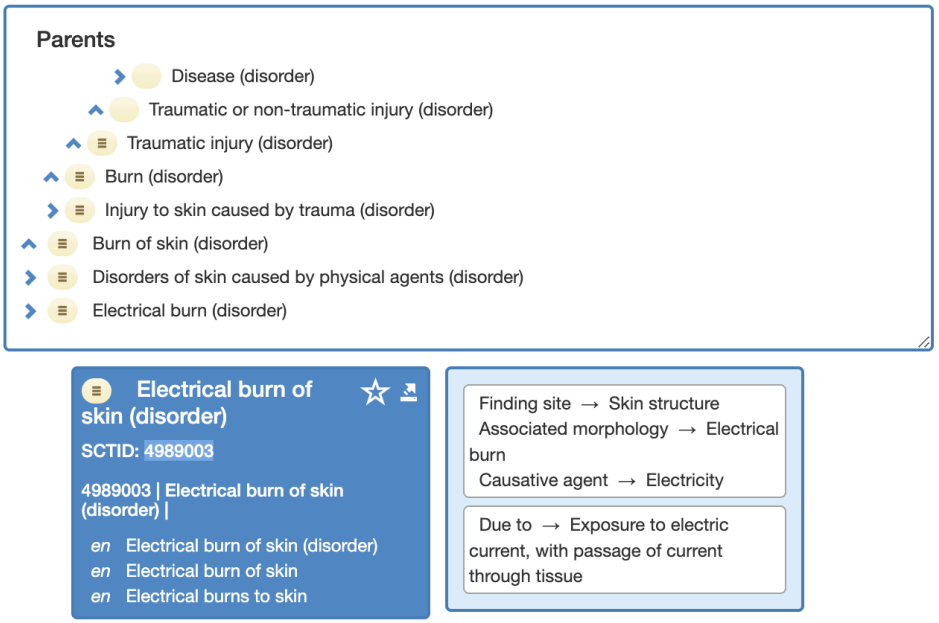

Figure 16. Relationship groups of the concept "*Electrical burn of skin (disorder)*".

[25] DisorderDescription: https://github.com/hacid-project/knowledge-graph/blob/main/ontologies/medical-dx/pattern/disorder.ttl
[26] Electrical burn of skin: https://snomed.info/id/4989003

Hence, the SNOMED-CT concept "*Electrical burn of skin (disorder)*" generates two instances of `mdx.DisorderDescription` that describes a single concept, i.e. `mdxd:4989003`[27] as exemplified in Figure 17. Those descriptions of disorder come from the two relationship groups available for Electrical burn of skin as presented in Figure 16, that is:

a) Finding site > Skint structure
   Associated morphology > Electrical burn
   Causative agent > Electricity
b) Due to > Exposure to electric current, with passage of current through tissue

The relationship groups a) and b) provide two descriptions of the same disorder. Those descriptions are then formalised with DnS as depicted in Figure 17.

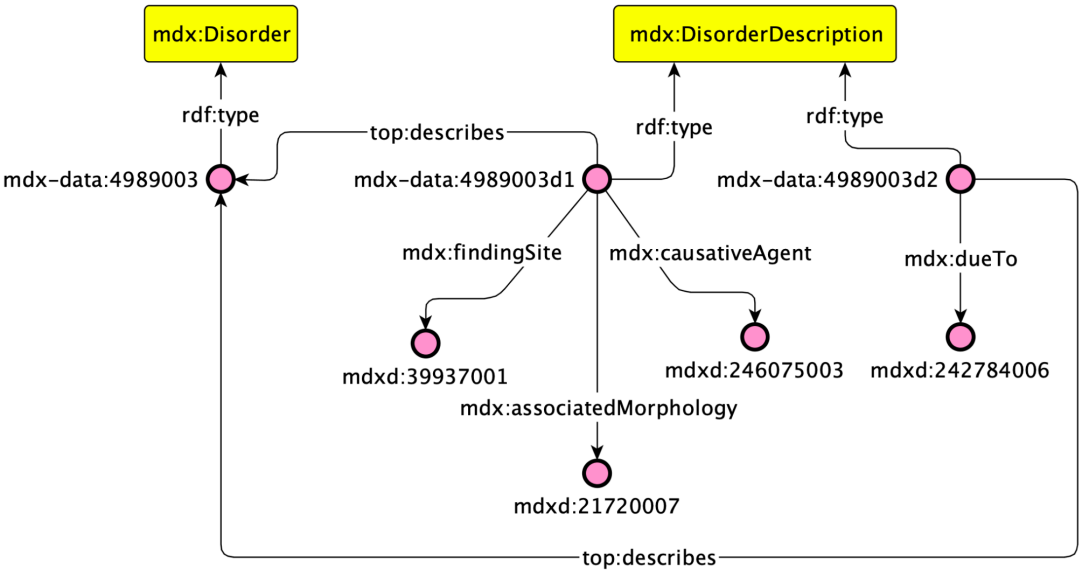

Figure 17. Descriptions of "*Electrical burn of skin (disorder)*".

The representation based on descriptions allows us to represent the occurrences of diseases in patients as situations, thus applying the DnS patterns. For example, a case like the following is converted to a representation like the one presented in Figure 18: *"Karl is diagnosed with electrical burns of skin due to exposure to electrical current."*

[27] The prefix `mdxd:` stands for the namespace `https://w3id.org/hacid/mdx/data/`.

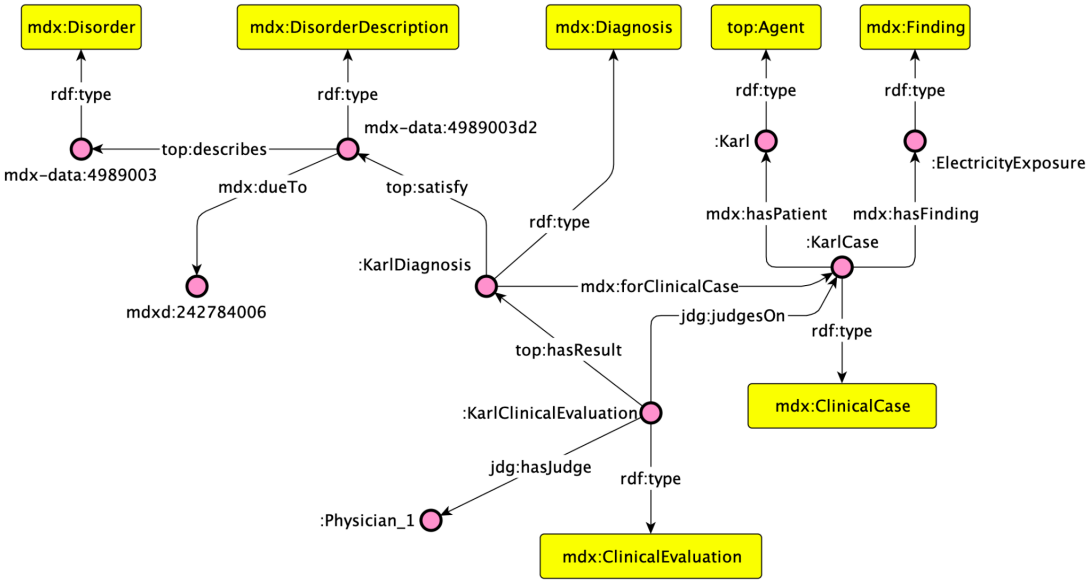

Figure 18. Example of DnS for medical diagnostics.

In the diagram in Figure 18 a `mdx:ClinicalCase`, i.e. `:KarlCase`, reports a finding of exposure to electricity, i.e. `:ElectricityExposure`, for the patient `:Karl`. This case is associated with a `mdx:ClinicalEvaluation`, i.e. `:KarlClinicalEvaluation`, that specialises the class `jdg:Judgement` from the Judgement module. The result of the judgement is a `mdx:Diagnosis`, which is the entity `:KarlDiagnosis`. The latter is `top:Situation` that satisfies a `top:Description`, which in turn is a certain description of Electrical burn of skin gathered from SNOMED-CT.
The DnS representation might enable neuro-symbolic reasoning on cases represented as situations that are required to be classified against descriptions of diseases. In fact, neuro-symbolic reasoning based on DnS has been recently introduced by a novel system called *Sandra* [34]. Sandra, given a set of facts (i.e. situations), infers all possible perspectives (i.e. descriptions) that can provide a plausible interpretation for it, even in the presence of incomplete information.
The descriptions from the domain concepts of SNOMED-CT and the domain knowledge graphs consisting of instances of descriptions and described clinical concepts were generated procedurally with a Python code. The latter is available as Jupyter notebook on the project's GitHub repository[28].

The following are the SPARQL queries used to test the CQs introduced in Table 4.

- mdx-1: *What are the specialties of a clinical case?*

```
PREFIX : <https://w3id.org/hacid/example-data/>
PREFIX mdx: <https://w3id.org/hacid/onto/medical-dx/>
SELECT ?specialty
WHERE {
```

[28] Jupyter notebook:
https://github.com/hacid-project/knowledge-graph/blob/main/code/HACID_MDX_Labels.ipynb

```
  :cx a mdx:ClinicalCase ;
        mdx:hasSpecialty ?specialty .
  ?specialty a mdx:Specialty
}
```

- mdx-2: *What are the diagnoses suggested by healthcare professionals?*

```
PREFIX : <https://w3id.org/hacid/example-data/>
PREFIX top: <https://w3id.org/hacid/onto/top-level/>
PREFIX mdx: <https://w3id.org/hacid/onto/medical-dx/>
SELECT ?diagnosis ?disorderDescr ?disorder
WHERE {
  ?evaluation a mdx:ClinicalEvaluation ;
        mdx:hasDiagnosis ?diagnosis ;
        mdx:hasDiagnostician ?agent .
  ?disorderDescr a mdx:DisorderDescription ;
                 mdx:isInScopeOfDiagnosis ?diagnosis ;
                 top:describes ?disorder .
  ?agent a mdx:HeatlhcareProfessional .
  ?disorder a mdx:Disorder
}
```

- mdx-3: *Which is the relevance order of a diagnosis provided by a certain healthcare professional?*

```
PREFIX : <https://w3id.org/hacid/example-data/>
PREFIX top: <https://w3id.org/hacid/onto/top-level/>
PREFIX jdg: <https://w3id.org/hacid/onto/core/judgement/>
PREFIX mdx: <https://w3id.org/hacid/onto/medical-dx/>
SELECT ?evaluation ?rank
WHERE {
  ?ranking a mdx:RankedClinicalEvaluation ;
           mdx:withRank ?rank ;
           mdx:ranks ?evaluation ;
           jdg:hasJudge :professional
}
```

- mdx-4: *What is the seniority, specialty, and organisation of a healthcare professional that performs a diagnosis?*

```
PREFIX : <https://w3id.org/hacid/example-data/>
PREFIX top: <https://w3id.org/hacid/onto/top-level/>
PREFIX ar: <https://w3id.org/hacid/onto/core/agentrole/>
PREFIX mdx: <https://w3id.org/hacid/onto/medical-dx/>
SELECT ?professional ?seniority ?specialty ?organisation
```

```
WHERE {
  ?professional a mdx:HeatlhcareProfessional .
  ?professionalRole a mdx:HeatlhcareProfessionalRole ;
                    ar:involvesAgent ?professional;
                    mdx:hasSeniority ?seniority ;
                    mdx:hasSpecialty ?specialty ;
                    mdx:worksFor ?organisation .
  ?evaluation a mdx:ClinicalEvaluation ;
              mdx:hasDiagnostician ?professional ;
              mdx:hasDiagnosis ?diagnosis .
  ?diagnosis a mdx:Diagnosis ;
             ^mdx:isInScopeOfDiagnosis/top:describes :viral-pneumonia
}
```

- mdx-5: *What is the disorder a diagnosis targets?*

```
PREFIX : <https://w3id.org/hacid/example-data/>
PREFIX mdx: <https://w3id.org/hacid/onto/medical-dx/>
SELECT ?diagnosis ?disorder
WHERE {
  ?diagnosis a mdx:Diagnosis ;
             ^mdx:isInScopeOfDiagnosis/top:describes ?disorder
}
```

- mdx-6: *What are the findings reported for a case?*

```
PREFIX : <https://w3id.org/hacid/example-data/>
PREFIX mdx: <https://w3id.org/hacid/onto/medical-dx/>
SELECT ?case ?finding
WHERE {
  ?case a mdx:ClinicalCase ;
          mdx:hasFinding ?finding
}
```

- mdx-7: *What are the solutions of a case?*

```
PREFIX : <https://w3id.org/hacid/example-data/>
PREFIX mdx: <https://w3id.org/hacid/onto/medical-dx/>
SELECT ?case ?evaluation
WHERE {
  ?case a mdx:ClinicalCase .
  ?evaluation a mdx:ClinicalEvaluation ;
              mdx:forClinicalCase ?case
}
```

3.4. DKG for climate services

3.4.1. Methods

As for the other parts of the ontology, also for the climate service case the eXtreme Design methodology is used, adapted to the specific context. One important aspect in which the design of the ontology for climate services differs from XD is that the scenario agreed with the domain experts was not sufficient to extract many of the relevant competency questions in the domain. Only through the discussion with the experts and the iterative refinement of the proposed ontology was it possible to gather much of the concepts and relationships. Possible reasons for that include the vastity of the domain and the inherent complexity of the problem solving process in the context of climate services.

3.4.2. Sources

The main sources of domain knowledge were the project deliverable D7.1 and the associated preparatory materials. In the deliverable, a climate service scenario is presented, along with the mockup of an application to manage a climate service workflow. Other sources of information have been identified from the discussions with domain experts that participate in the project.

3.4.3. Competency Questions

The competency questions we present are obtained as a result of many iterations with domain experts.

Table 7. Competency questions for climate services modelling.

ID	**Competency question**
cs-1	What are the climate cases represented in the knowledge graph?
cs-2	What is a specific case about (description, relevant time/space scope, type of assessment required, optionally accepted risk level)?
cs-3	What are the relevant hazard types for a case?
cs-4	What specific questions have been associated with a case?
cs-5	What answers have been given to a case or specific question?
cs-6	What information does an answer provide in terms of identifying relevant projections (e.g., adopted model, available variable, specific projection, …)?
cs-7	Which known projections are compatible with a given answer?
cs-8	What are the known models (of a certain type)?

3.4.4. Ontology Modules

The ontology designed for climate services is presented as a set of modules concerning different aspects of the domain.

Note on terminology and graphical representation. As for the other ontologies presented, Graffoo is used to visually represent the OWL ontology. The prefix `cs:` is used in the text to refer to the namespace corresponding to the climate services ontology. In the graphical representation, for reasons of space, such namespace is considered as base URI, so that there is no need to use the prefix.

Models, Simulations, and Projections

A central aim of climate science is trying to predict future climate conditions, based on past and present data and adopting some model for the considered system. The models (class `cs:Model`) considered here are algorithms that represent the behaviour and evolution of a system in time. The models considered are often climate models (`cs:ClimateModel`), i.e., representing climate dynamics in physical terms, but socioeconomic models like IAMs (Integrated Assessment Models, class `cs:IntegratedAssessmentModel`) are also of interest when dealing with future collective human behaviour.

Different climate models are used for different spatial coverages/granularities: a `cs:GlobalClimateModel` is one that encompasses the whole earth surface, usually with coarse granularity (e.g., 60 Km); a `cs:RegionalClimateModel` is one that encompasses only a part of the world (e.g., Europe) but is more fine grained (e.g., 12 Km); a `cs:LocalClimateModel` is one that encompasses a yet smaller area (e.g., the United Kingdom), with still more detail (e.g., 2 Km). A special case of local climate models are convection permitting models (class `cs:ConvectionPermittingModel`), i.e. models with enough detail to explicitly represent convection.

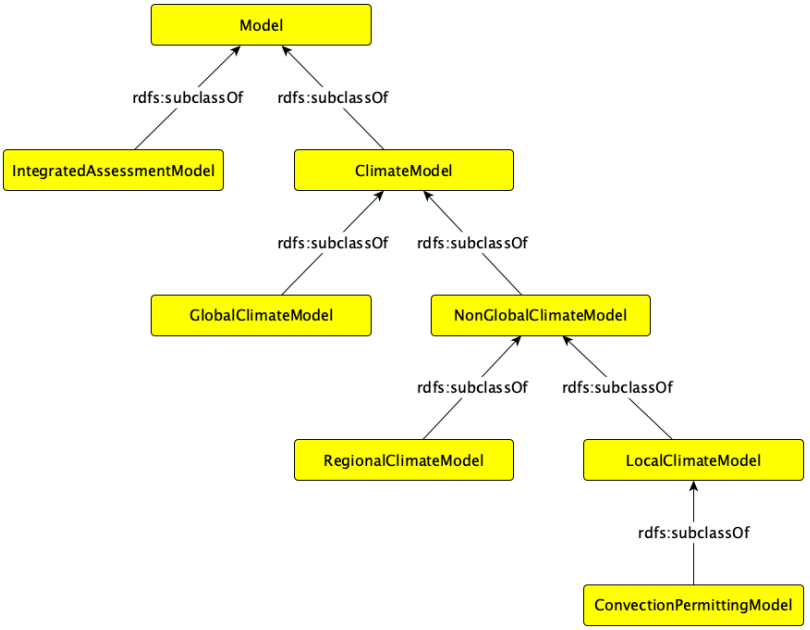

Figure 19: Models employed in climate services.

Simulations and Projections

The execution of a model is a simulation, as it simulates the behaviour of some (real) system. Such (model-based) simulation (`cs:ModelBasedSimulation`) is a special case of a `cs:DataTransformation`, i.e. the execution of a computing process whose (main) purpose is to get some output data from some input data (no assumptions are made about determinism or the absence/presence of side effects). It should be stressed that an instance of `cs:DataTransformation` is a single execution of such transformation in time/space on some computing device, albeit these details do not necessarily need to be represented.

The ontology supports the common use case in which an ensemble of simulations is run (`cs:EnsembleModelBasedSimulation`) distinguishing it from the single-simulation case (`cs:SingleModelBasedSimulation`). It should be stressed that the single/ensemble characterisation refers to the simulation, not the model: an ensemble simulation can be composed of executions of different models, but also executions of the same model with different parameters (or even the same parameters, which may be useful in the case of a stochastic model).

Simulations are further classified according to the type of model: `cs:IAMSimulation` when it is an `cs:IntegratedAssessmentModel`, `cs:ClimateModelSimulation` when it is a `cs:ClimateModel`, `cs:GCMSimulation` when it is a `cs:GlobalClimateModel`. The inputs of a `cs:ModelBasedSimulation` always include the adopted model (or possibly models, if it is an `cs:EnsembleModelBasedSimulation`) plus additional constraints depending on the kind of model/simulation: an `cs:IntegratedAssessmentModel` takes as input a `cs:Pathway`, i.e. a set of coarse socio economic or emission assumptions that is used to drive such simulation; a `cs:GlobalClimateModel` takes as input an `cs:EmissionScenario`, i.e. a specific evolution of emissions in time that is often the output of an `cs:IntegratedAssessmentModel`.

Climate model simulations generate as output projections (`cs:ClimateProjection`) which are, as the emission scenarios, spatiotemporal datasets (further described below).

A specific case of a `cs:ClimateModelSimulation` (not being a `cs:GCMSimulation`) is a `cs:DynamicalDownscaling`, i.e. when a simulation is constrained by the output of an upscale simulation (e.g., a projection obtained by a global climate model simulation is used as input to a regional climate model simulation). `cs:ProjectionDownscaling` is the generalisation of `cs:DynamicalDownscaling` to processes that are not simulations, for example the downscaling of a projection by statistical means (`cs:StatisticalDownscaling`).

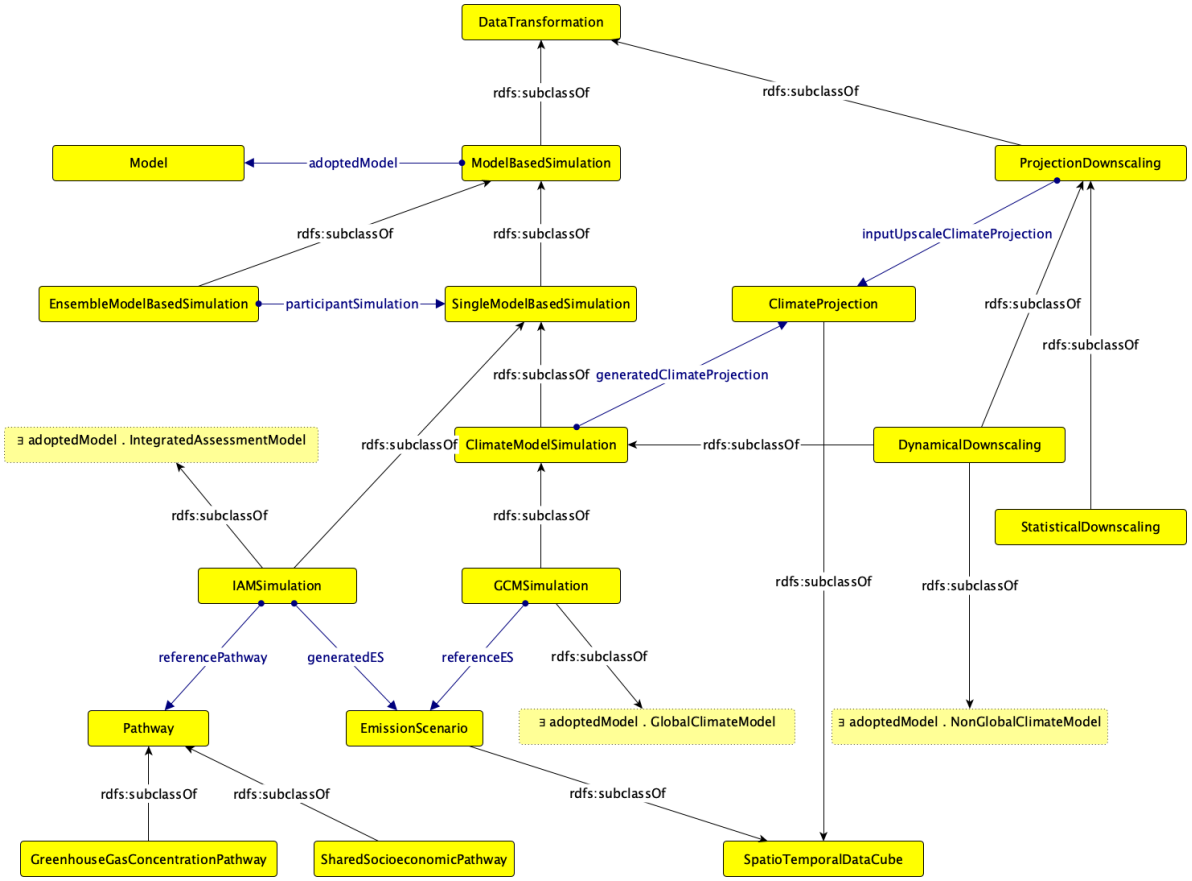

Figure 20: Ontology module for climate simulations and projections.

Datasets

Climate experts deal with spatiotemporal datasets in a number of roles: to represent historic climate data, outputs of climate models (projections), expected emission scenarios, etc. To represent these datasets, the RDF Data Cube vocabulary is adopted and specialised. Terms from the Dublin Core vocabulary are adopted as well.

The RDF Data Cube vocabulary distinguishes between a `qb:Dataset` and its associated `qb:DatastructureDefinition` (which can be shared across multiple datasets). The structure of a dataset is represented by associating the corresponding `qb:DatastructureDefinition` with a set of component specifications (class `qb:ComponentSpecification`), each one of them having the role of either a dimension, a measure, or an attribute.

A `cs:SpatioTemporalDataCube` is a specialisation of a `qb:Dataset` that has specific spatiotemporal coverage (`dcterms:spatial` and `dcterms:temporal`) and its structure contains at least a temporal dimension and a spatial dimension. The measure components of a `cs:SpatioTemporalDataCube` correspond with variables (class `cs:Variable`).

Examples of simple climate variables include maximum temperature and average humidity. A special subgroup of variables are indices (class `cs:Index`), i.e. derived variables that can be computed from simple ones.

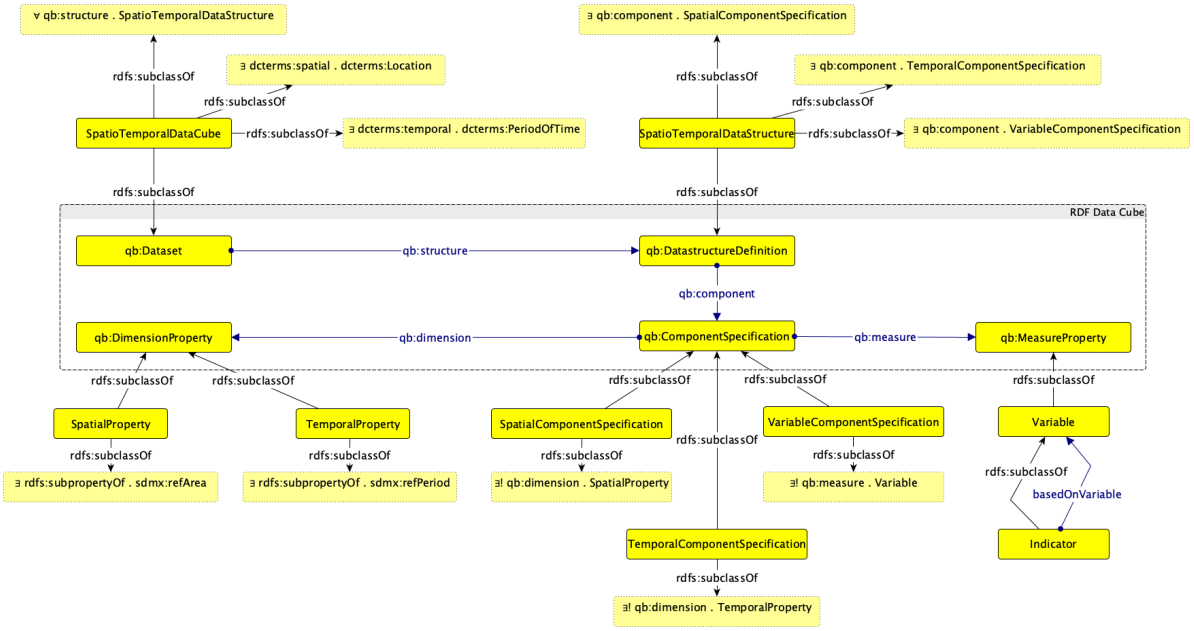

Figure 21: Ontology module for datasets.

Climate Case and Contributions

Climate service provisioning revolves around specific climate cases (`cs:ClimateCase`) that are requests from a customer to the climate service provider. Relevant aspects of a climate case are the spatial and temporal (usually in the future) interval of interest, types of hazard that need to be considered, and accepted risk level.

The cooperative analysis of a case involves an exchange of informative contributions (`cs:AssessmentContribution`) between multiple agents, which are classified as either requests of information (`cs:Request`) or assessments (`cs:Assessment`). These contributions may be related to the whole case or specific aspects and the `cs:repliesTo` property associate assessments to the specific requests they respond to.

A climate case is itself a request (from the customer to the climate service provider) and may have a hierarchy of subrequests on specific topics.

The class `cs:AssessmentType` enables the classification of request types using appropriate controlled vocabularies, which may be either generic or specific to a climate service provider.

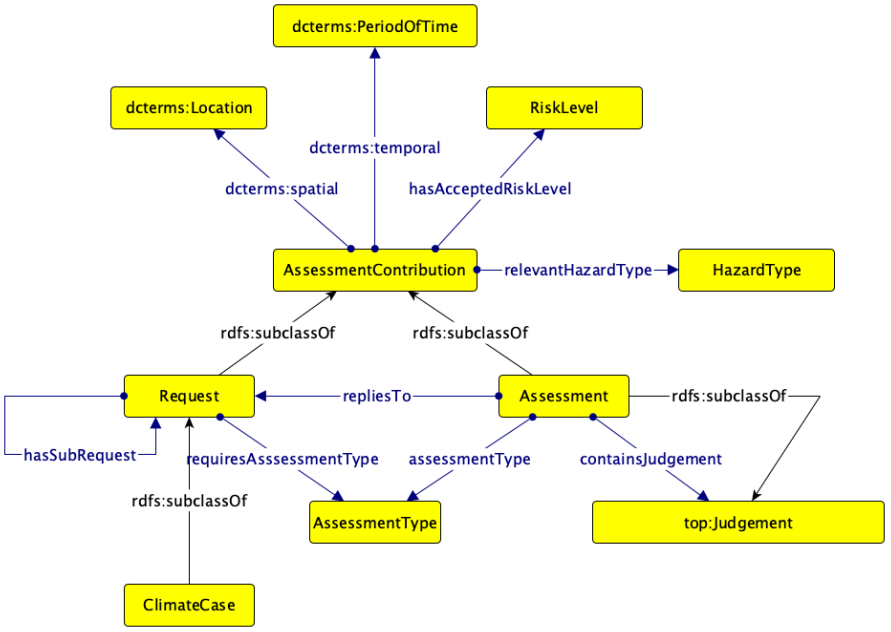

Figure 22: Ontology module for climate case and contributions.

Projection Constraints

A fundamental aspect in the analysis of a climate case is finding, choosing, and evaluating relevant climate projections. A type of judgement that is especially useful is hence suggesting some specific projections or characteristics that projections on interest should have. The results of these judgments can thus be represented as constraints on the set of relevant projections (`cs:ProjectionConstraint`). Such judgements are called `cs:ProjectionConstraintJudgement`).

Rather than defining ex novo a vocabulary to specify all the constraints that may be of interest, the Shapes Constraint Language (SHACL)[29] vocabulary is reused. The SHACL class `sh:NodeShape` represents a constraint on a set of resources, which in our usage is a set of potential candidate projections relevant to the case. The specific constraint is defined through the usage of multiple SHACL properties and can be quite complex. While for complete description we defer the reader to the SHACL documentation, some simple constraints are the following ones: `sh:hasValue`, constrains the resource to be a specific value; `sh:in`, constrains the resources to be members of a specific set; `sh:class`, constrains the resources to be a certain type. A `cs:ProjectionConstraint` is a `sh:Shape` that always includes a `sh:class` constraint with value `sh:Projection`, i.e. it must identify projections.

[29] https://www.w3.org/TR/shacl/

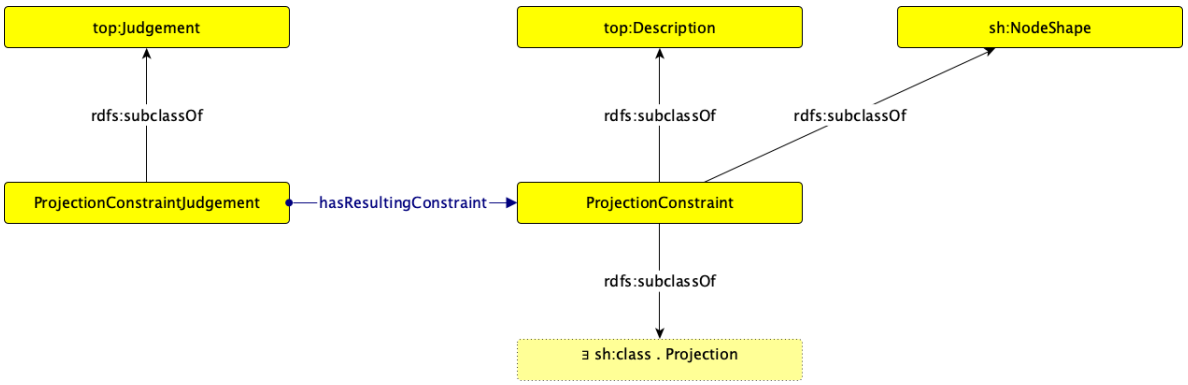

Figure 23: Module for representing projection constraints.

Climate Phenomena

Climate scientists classify and study climate phenomena, from large scale processes as oscillations to small scale events like a local flooding. The ontology offers a way to represent climate phenomena (class `cs:ClimatePhenomenon`) and to classify them by type (`cs:ClimatePhenomenonType`) using controlled vocabularies.

Phenomena can sometimes be characterised with indexes (for example WMO defines heatwaves as occurring when the daily maximum temperature of more than five consecutive days exceeds the average maximum temperature by 5 °C, the normal period being 1961–1990). For that purpose the class `cs:PhenomenonCharacterisation` is defined. Hazards (class `cs:Hazard`) are in this context a specific subclass of phenomena, namely the ones that may potentially cause harm to people and property, directly or indirectly.

Furthermore, it is of interest to model the interaction between phenomena. The property `cs:relatedWithPhenomenon` is a symmetric transitive property that describes a generic relationship between two phenomena. Subproperties of `cs:relatedWithPhenomenon` describe more specific relationships: `cs:featuresPhenomenon` relates a complex phenomenon with phenomena that are part of it (e.g., the “El Niño–Southern Oscillation” features the “1997–1998 El Niño event”); `cs:causedPhenomenon` relates a phenomenon with another that is considered a cause of the former one; `cs:influencesPhenomenon` relates a phenomenon with another that ha been influenced by the former one during its time of existence. All said properties are transitive.

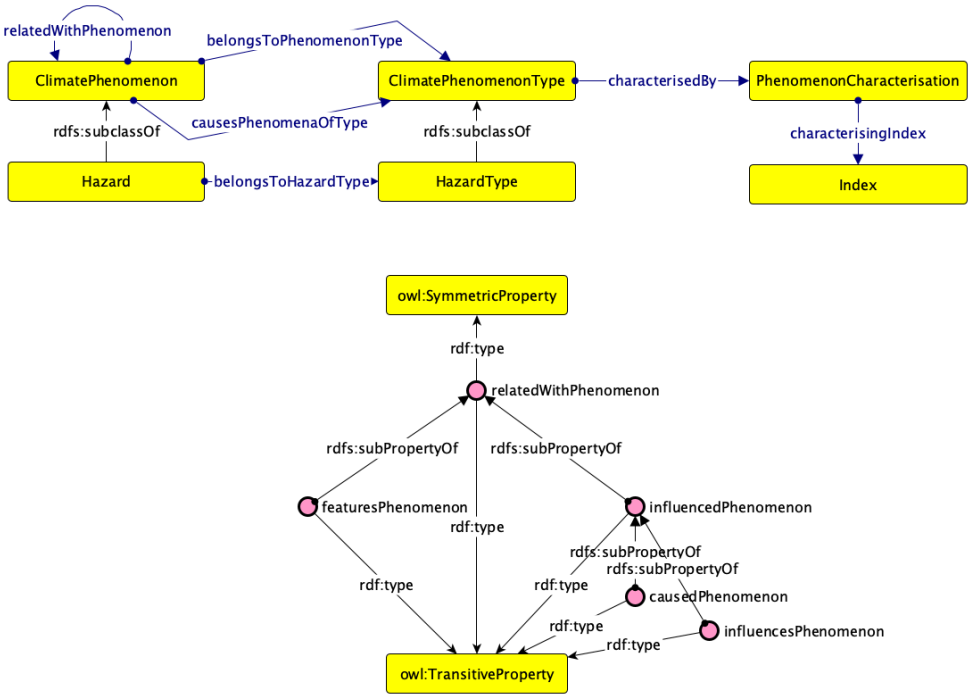

Figure 24: Module specifying climate phenomena and hazards.

3.4.5. Competency Questions Representation

In order to validate this part of the ontology, here the competency questions (CQs) defined in 3.4.3 are represented as SPARQL queries over the representation presented in 3.4.4. Where the results depend on some specific resources of the knowledge graph (for example, the case considered), we provide a parametric SPARQL query, i.e. a query with placeholders for parameters. Such parameters will be represented as SPARQL variables with the notation $..., while variables not meant to be used as parameters will follow the notation ?...

CQ cs-1: What are the climate cases represented in the knowledge graph?

```
PREFIX cs: <https://w3id.org/hacid/onto/cs/>
SELECT ?case
WHERE {
  ?case a cs:ClimateCase
}
```

CQ cs-2: What is a specific case about (description, relevant time/space scope, type of assessment required, optionally accepted risk level)?

Parameters:

- `$case` of type `cs:ClimateCase`

```
PREFIX cs: <https://w3id.org/hacid/onto/cs/>
```

```
PREFIX dcterms: <http://purl.org/dc/terms/>
SELECT ?description ?spaceDomain ?temporalDomain ?assessmentType
WHERE {
  $case dcterms:description ?description;
        dcterms:spatial ?spatialScope;
        dcterms:temporal ?temporalScope;
        cs:assessmentType ?assessmentType.
  OPTIONAL {$case cs:hasAcceptedRiskLevel ?acceptedRiskLevel}
}
```

CQ cs-3: What are the relevant hazard types for a case?

Parameters:

- **$case** of type `cs:ClimateCase`

```
PREFIX cs: <https://w3id.org/hacid/onto/cs/>
SELECT ?hazardType
WHERE {
  $case cs:relevantHazardType ?hazardType.
}
```

CQ cs-4: What specific questions have been associated with a case?

Parameters:

- **$case** of type `cs:ClimateCase`

```
PREFIX cs: <https://w3id.org/hacid/onto/cs/>
SELECT ?subRequest ?requestedAssessmentType
WHERE {
  $case cs:hasSubRequest+ ?subRequest.
  ?subRequest cs:requiresAssessmentType ?requestedAssessmentType.
}
```

CQ cs-5: What answers have been given to a case or specific question?

Parameters:

- **$question** of type `cs:Request`

```
PREFIX cs: <https://w3id.org/hacid/onto/cs/>
SELECT ?answer ?answerAssessmentType
WHERE {
  ?answer cs:repliesTo $question;
          cs:assessmentType ?answerAssessmentType.
}
```

CQ cs-6: What information does an answer provide in terms of identifying relevant projections (e.g., adopted model, available variable, specific projection, …)?

The set of relevant projections is identified through constraints using the SHACL vocabulary. It would be cumbersome (if at all possible) to express the identification of a SHACL constraint in a single SPARQL query.
The following parameter query is just an example, it returns constraints associated with an answer and is able to identify the three specific types of constraints mentioned in the CQ.

Parameters:

- `$answer` of type `cs:Assessment`

```
PREFIX sh: <http://www.w3.org/ns/shacl#>
PREFIX cs: <https://w3id.org/hacid/onto/cs/>
SELECT ?constraint ?requiredProjection ?requiredModel ?requiredVariable
WHERE {
  $answer cs:containsJudgement ?constrJudgement;
  ?constrJudgement a cs:ProjectionConstraintJudgement;
                   cs:hasResultingConstraint ?constraint.
  OPTIONAL {?constraint sh:hasValue ?requiredProjection}
  OPTIONAL {
      ?constraint sh:property [
        sh:path (
          [sh:inversePath cs:generatedClimateProjection]
          cs:adoptedModel
        );
        sh:hasValue ?requiredModel
      ]
  }
  OPTIONAL {
      ?constraint sh:property [
        sh:path (qb:structure qb:component qb:measure);
        sh:hasValue ?requiredVariable
      ]
  }
}
```

CQ cs-7: Which known projections are compatible with a given answer?

As mentioned above, many different types of constraints are possible in SHACL and they cannot be expressed in a single query. In order to find all the projections compatible with some given constraints a general mechanism has to be built. This could be based on existing SHACL libraries and it will involve generating one or more SPARQL queries dynamically.
We show an example query based on one of the three specific cases considered in CQ 37 (the constraint on the variable).

```
PREFIX sh: <http://www.w3.org/ns/shacl#>
PREFIX cs: <https://w3id.org/hacid/onto/cs/>
```

```
SELECT ?projection
WHERE {
  $answer cs:containsJudgement ?constrJudgement;
  ?constrJudgement a cs:ProjectionConstraintJudgement;
                   cs:hasResultingConstraint ?constraint.
  OPTIONAL {?constraint sh:hasValue ?requiredProjection}
  OPTIONAL {
      ?constraint sh:property [
        sh:path (qb:structure qb:component qb:measure);
        sh:hasValue ?requiredVariable
      ]
      ?projection qb:structure/qb:component/qb:measure ?requiredVariable
  }
}
```

CQ cs-8: What are the known models (of a certain type)?

Parameters:

- `$modelClass` subclass of `cs:Model`

```
PREFIX cs: <https://w3id.org/hacid/onto/cs/>
SELECT ?model
WHERE {
  ?model a $modelClass.
}
```

3.5. Knowledge Graph Population and logical validation

The knowledge graph on medical diagnostics counts of 22,033,693 triples and can be queried on a Virtuoso instance[30] via SPARQL. It was populated with the instances of all clinical terms gathered from SNOMED-CT, i.e. each class of the conceptual model (cf. Section 3.2) was populated with the instances that specialises such a concept in SNOMED-CT. Hence, the property groups were used for instantiating individuals of the available subclasses of `top:Descriptions`, e.g. `mdx:DisorderDescription`, `mdx:BodyStructureDescription`, etc. Whenever possible, the individuals were aligned with Wikidata by relying on the mapping provided by Wikidata between its entities and SNOMED clinical terms.

We also included alternative namings, i.e. the Naming core module, for clinical terms by reusing a Meta-inventory [35], which is a database of medical abbreviations and acronyms for natural language processing. Currently, we are investigating a sound solution to identify boundaries around entities about medical terminology in Wikidata to enrich our knowledge graph with additional data.

[30] The address of the SPARQL endpoint will be provided to reviewers privately as we follow the same data policy associated with SNOMED-CT as our KG contains a significant subset of SNOMED-CT.

The logical assessment of the KG was performed by using the alignment with DOLCE for enabling DL reasoning and validating consistency and coherency of the KG. The functional assessment was carried out through the CQs once converted to SPARQL queries.

4. Experiments with bottom-up KG construction

In this section, we provide an experimental evaluation of the proposed bottom-up KG construction methods, to determine the expected performance of the approach before utilising it in the medical diagnostics and climate service scenario. For medical diagnostics, we intend to extract data from sources of evidence (e.g., scientific papers), in order to ground the knowledge available in the engineered DKG on trustworthy resources. Regarding the climate services segment, our intention is to employ these strategies to populate the DKG especially for all those entities and relations that cannot be found in the available sources. Specifically, due to the scarcity of parallel datasets in climate services, the bottom-up paradigm is envisioned as the primary method for constructing the DKG.

4.1. Data

We utilise the ADE [36] (Adverse Drug Effect) dataset for our preliminary experiments. Developed as a benchmark corpus, the ADE dataset facilitates the automatic extraction of adverse effects associated with drugs from medical case reports. It serves as a foundation for training and validating models aimed at classifying informative and non-informative sentences in the realm of drug-related safety concerns. The dataset comprises systematically double-annotated medical case reports, which are subsequently harmonised to create representative consensus annotations.
The ADE dataset encompasses a total of 420,515 tokens across 20,967 sentences. Among these sentences, 4,272 have been annotated to include information regarding the names and relationships between drugs, adverse effects, and dosages. In our current experiments, we leverage all the annotated sentences, aligning 10,839 entities (comprising 5,063 drugs and 5,776 adverse effects) and 6,821 relations (pertaining to drug-adverse effect relationships). An example of a parsed sentence and triplets pair of ADE dataset is provided below:

Sentence: *We report a case of fulminant hepatic failure associated with didanosine and masquerading as a surgical abdomen and compare the clinical , biologic , histologic , and ultrastructural findings with reports described previously .*

Entity triplet: *(fulminant hepatic failure ; instance of ; disease) (didanosine ; instance of ; drug)*

Relation triplet: *(fulminant hepatic failure ; effect ; didanosine)*

Here we use the ADE dataset because (i) ADE is a widely used and annotated dataset of medical diagnostics in NER and relation extraction, and (ii) ADE functions as a smaller and experimental dataset representative of the real-world diagnostic cases used within HACID

and the domain knowledge represented in the DKG, facilitating preliminary investigations into the feasibility of our methodologies. Thus, we conduct preliminary experiments on ADE to validate the efficacy of the approaches under consideration.

4.2. Neural KG generation

We tuned a GPT-2 model with ADE dataset using the prefix-tuning method [37] and leverage the DEEPSTRUCT model as a validator to validate the generated triplets. The prefix-tuning method is introduced as a means to alleviate the temporal and financial costs associated with the fully fine-tuning approach. In contrast to fully fine-tuning a pre-trained model, which entails updating all parameters within the neural network model, the prefix-tuning method involves updating only a limited set of additional parameters while maintaining the pre-trained model in a frozen state.

Pipeline

Our approach to knowledge construction is described in Figure 25. The whole pipeline contains two major parts: generator and validator, fulfilling the process of prefix-tuning.
The generator is a generative model that is intended to be prefix-tuned on vanilla LLM (in this case, GPT-2) with additional parameters (H0). Originally, these additional parameters are free parameters that are initialised randomly. In our case, we tried to leverage these parameters by adding pre-defined ontologies, or relation types, while the backbone LLM remains the same during prefix-tuning by fixing the parameters. More specifically, the prefix (e.g. *relation:effect*) is vectorised and fed into these additional learnable parameters, following which the outputs are prepended to input sentence vectors, which are fed into the frozen autoregressive LLM for the further tuning.
For the validator, we simply use the DEEPSTRUCT model because it is already trained on multiple datasets for multiple knowledge relevant tasks, including ADE. During the prefix-tuning process, the triplets generated by DEEPSTRUCT will be acting as the targets for computing the loss between them and the ones predicted by the generator.

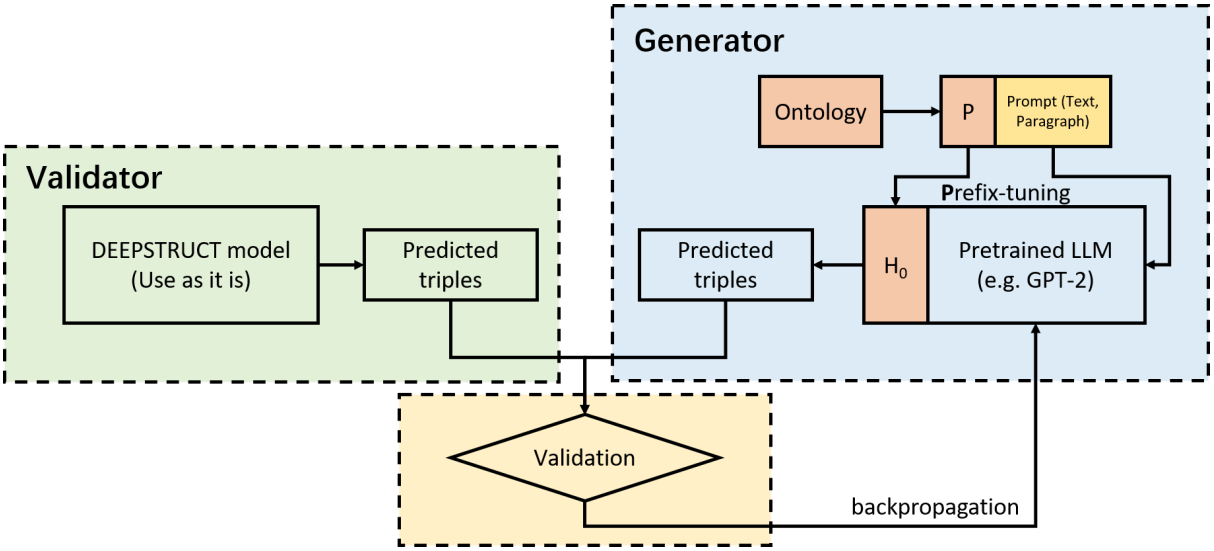

Figure 25. The pipeline for DKG construction by the neural KG generation paradigm.

Experiment and results

Two experiments were conducted to assess the efficacy of the proposed approach in both tasks: Named Entity Recognition (NER) and relation extraction (REL). And for each task, we conduct our pipeline on two folds of the ADE dataset, ade0 and ade1, respectively and independently. Each fold comprises 3,845 pairs of case description sentences and their corresponding triplets, while the test set comprised 427 pairs. The experimental results are presented in Table 8 below.

Regarding NER, our approach exhibited marginally lower performance compared to utilising solely the DEEPSTRUCT model (DS). The F1 scores of our approach are around 0.87 while DEEPSTRUCT provides 0.9 on ade0 and 0.99 on ade1. Regarding REL, we observe similar results, although the F1 scores of our approach range from 0.66 to 0.7, while DEEPSTRUCT provides 0.8 on ade0 and 0.98 on ade1. The outcome can be attributed to two potential factors. Firstly, DEEPSTRUCT's pre-training utilises the entire ADE dataset, thereby having already acquired knowledge of all ADE dataset triplets. Additionally, we employed the triplets generated by DEEPSTRUCT as the target rather than the original annotated triplets. Nevertheless, it is imperative to conduct experiments without relying solely on the original annotated triplets, given the scarcity of well-annotated datasets.

However, DEEPSTRUCT functions as a pre-trained KG model, limiting its ability to recognise or extract triplets to specific entity types and relation types predefined within the dataset on which it was trained (e.g., ADE comprising 2 entity types and 1 relation). This inherent limitation may impede the flexibility and evolution of the ontology design within the HACID project.

Table 8. The results of DKG construction in fine-tuning paradigm.

	NER (ade0)		NER (ade1)		REL (ade0)		REL (ade1)	
	Ours	DS	Ours	DS	Ours	DS	Ours	DS
Precision	0.8715	0.9272	0.8843	0.9973	0.5829	0.8214	0.6695	0.9871
Recall	0.8577	0.8830	0.8741	0.9883	0.7673	0.7984	0.7373	0.9913
F1 Score	0.8645	0.9046	0.8791	0.9928	0.6626	0.8097	0.7018	0.9892

4.3. RAG paradigm

We also implemented a RAG system by leveraging LlamaIndex[31] and deployed an experiment to test the OIE performance on the ADE dataset. Basically, the RAG process contains two stages: 1) indexing and building a knowledge graph, 2) querying indices and generating results using LLM.

KG-based RAG system

Figure 26 presents an overview of an RAG system. During the indexing stage, both structured data sources (such as databases) and unstructured data sources (such as documents) are subjected to indexing procedures. This involves parsing documents into discrete segments (or chunks), from which knowledge is extracted in the form of triplets,

[31] https://docs.llamaindex.ai/en/stable/

formatted as (entity, relation, entity). Subsequently, this extracted knowledge is embedded and organised into a KG to facilitate subsequent utilisation. In our experiments and project, we can particularly utilise the triplets that have already been provided as a KG, thereby obviating the need for further information extraction.
During the querying stage, the LLM is prompted through a specific query, for instance, "*Extract triplets from the provided sentence*", to generate the triplets, accompanied by additional information retrieved from the indexed documents and the KG.

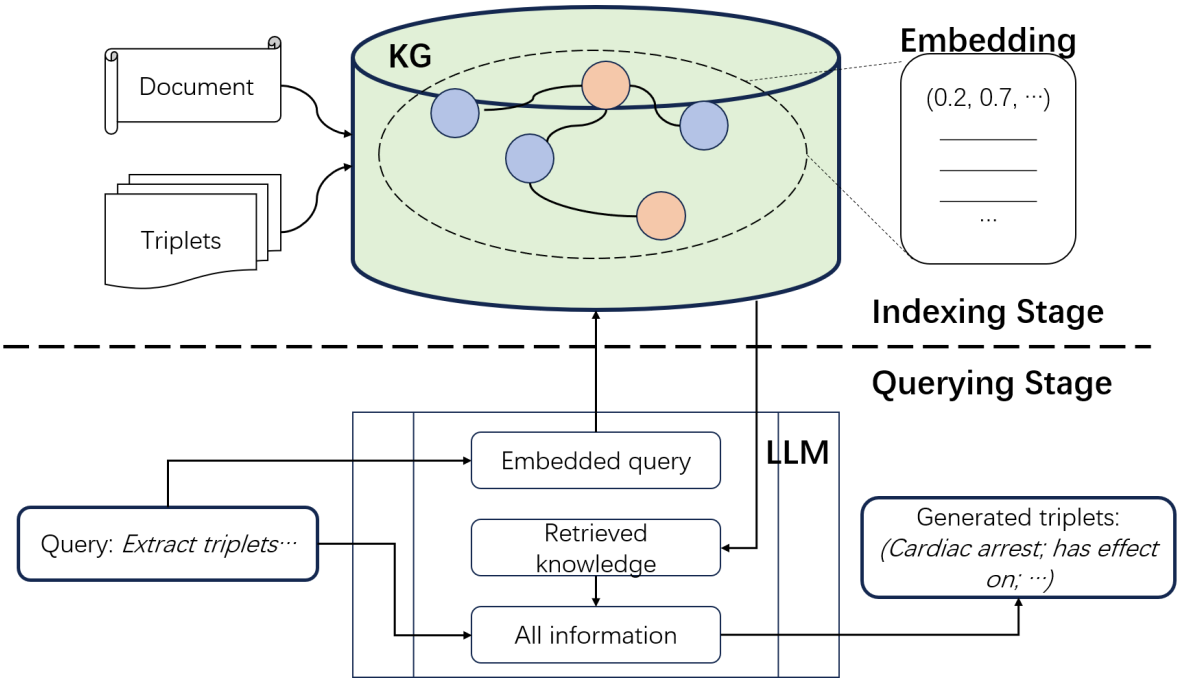

Figure 26. The pipeline for DKG construction in RAG paradigm.

Preliminary Experiment

As ADE provides both case description sentences as free text (regarded as the documents in Figure 26) and aligned annotated triplets, it is possible to do indexing by utilising both. Indeed, we expect that the usage of RAG within the HACID project can exploit available structured knowledge from the deployed DKG without a corresponding text. Hence, in these experiments we aim at testing RAG when only knowledge structured as triplets is provided. We deploy three settings of source of data in the experiments.

1. Indexing with documents and triplets, querying with documents and triplets.
2. Indexing with only triplets, querying with triplets.
3. Skipping indexing, querying with triplets.

We have not considered a setting in which indexing is performed with both documents and triplets while querying only uses triplets because the embeddings for the triplets obtained from the indexing stage remain the same regardless of whether or not documents are provided. In other words, the embeddings of the triplets are independent from the ones of the documents.
Additionally, in the querying stage, we provide the RAG system with different combinations of entities and relations within the prompt. There are three types of combinations we use for query:

1. only provide the target relation (referred to as *only_rel*);
2. provide the subject and the target relation (referred to as *sub+rel*);
3. provide the target relation and the object (referred to as *rel+obj*).

Here is an example for the query for the prompt of the *only_rel* approach:

```
{sentence}, Generate triplets using the sentence above with the format of
(subject ; {relation} ; object). For example, (cyclophosphamide ; has
adverse effect ; hemorrhagic cystitis)
```

In this query command, `{sentence}` denotes the case description sentence, and `{relation}` denotes the desired relation between entities. For the ADE experiment, we always set the relation to "*has adverse effect*". These two parameters are provided while `subject` and `object` are the one presumed to be generated. In addition, we establish an example `(cyclophosphamide ; has adverse effect ; hemorrhagic cystitis)` as a constraint defining the type of desired entities and triplet format. This exemplifies a few-shot learning strategy within prompt engineering methodologies.

Results

The results of the preliminary experiments with different settings are presented in Table 9 below. In the tabulated results, Experiment Setting 1 demonstrates the highest overall performance. Settings 2 yields results closely comparable to Setting 1, with precisions marginally lower than Setting 1 and recalls similar to or higher than Setting 1. Conversely, Setting 3, which exclusively employs triplets during the query stage, exhibits the lowest scores. Regarding query types, it is observed that *rel+obj* yields the highest scores, followed by *sub+rel* and *only_rel*.

Table 9. The results of DKG construction in RAG paradigm.

	Setting 1			Setting 2			Setting 3		
	only_rel	sub+rel	rel+obj	only_rel	sub+rel	rel+obj	only_rel	sub+rel	rel+obj
Hit triples	412	550	595	403	548	610	380	474	550
Gen triples	694	697	718	703	718	761	724	698	734
Target triples	696	696	696	696	696	696	696	696	696
Precision	0.594	0.789	**0.829**	0.573	0.763	0.802	0.525	0.679	0.749
Recall	0.592	0.79	0.855	0.579	0.787	**0.876**	0.546	0.681	0.79
F1 score	0.593	0.79	**0.842**	0.576	0.775	0.837	0.535	0.68	0.769

In summary, the embeddings of triplets significantly enhance the outcomes of the RAG, as evidenced by the comparison among Setting 1 (0.593-0.79-0.842), Setting 2 (0.576-0.775-0.837), and Setting 3 (0.535-0.68-0.769) according to the F1 scores. The performance difference between Setting 1 and Settings 2 is small, implying that document context is less pivotal compared to embeddings in querying a KG-based RAG system. Moreover, regarding query types, it is rational that greater information input (such as *sub+rel* and *rel+obj*) yields superior performance compared to solely providing relations. For example, with Setting 1, only providing a relation receives the lowest F1 score (0.593) and providing one more element increases F1 score significantly (0.79 and 0.842). To enhance the performance of the *only_rel* scenario, integrating existing NER models as an additional module within the RAG system during query and generation stages is recommended. It is

also worthwhile to explore more specific prompt instructions, through some prompt engineering approach.

4.4. Comprehensive comparison and Conclusions

To conduct a comprehensive comparison across paradigms, we re-organise the results of the experiments in section 4.2 and 4.3 and show them in Table 10.

Table 10. The comprehensive results of DKG construction across paradigms.

	RAG (Setting 2)			Prefix-tuning		DEEPSTRUCT	
	only_rel	sub+rel	rel+obj	REL(ade0)	REL(ade1)	REL(ade0)	REL(ade1)
Precision	0.573	0.763	0.802	0.5829	0.6695	0.8214	0.9871
Recall	0.579	0.787	0.876	0.7673	0.7373	0.7984	0.9913
F1 score	0.576	0.775	0.837	0.6626	0.7018	0.8097	0.9892

Our objective is to generate triplets for constructing a KG, wherein entities are interconnected by relations. For the prefix-tuning paradigm, we have opted for the results of REL from both our prefix-tuning pipeline and the original DEEPSTRUCT model. In the RAG paradigm, we have selected the results from setting 2 as they closely align with our subsequent HACID project.

As illustrated in the table, RAG's *only_rel* exhibits a lower F1 score (0.576) compared to prefix-tuning, while *sub+rel* and *rel+obj* (0.775 & 0.837) surpass prefix-tuning by more than 10% F1 score difference. This observation suggests that with external support, RAG can achieve significantly better outcomes than the tuning paradigm (0.6626 & 0.7018), approaching or surpassing some results of DEEPSTRUCT (0.8097 & 0.9892). Hence, as previously mentioned, integrating existing NER models as an additional module and employing prompt engineering strategies hold promise.

In the context of HACID, information extraction serves as the initial step, necessitating the expansion of the existing KG through the generation of new knowledge (i.e., new triplets). Thus, leveraging the generative capabilities of Language Model Models (LLMs) becomes imperative. In the RAG paradigm, integrating LLMs is straightforward and interchangeable as a module, whereas in the tuning paradigm, language models require training or fine-tuning, which is both time-consuming and costly. Moreover, the project's dataset updating requirements must be taken into account. Updating the dataset in the RAG paradigm is more flexible than retraining on an entirely new dataset.

Considering these factors, in the subsequent stages of the HACID project, RAG is anticipated to be the focal point for KG construction, while fine-tuning methods can also be employed in independent modules.

5. Conclusion

This document delves into the methodologies behind crafting and constructing the domain knowledge graph (DKG) – the cornerstone of the HACID project's hybrid human-artificial

collective intelligence framework. The DKG serves as a repository of domain-specific knowledge for the project's two chosen case studies: medical diagnostics and climate change adaptation management, also known as climate services. During the following activities of harmonisation and validation of the DKG, we will exploit the flexibility and modularity of the approach to propose adaptations and extensions that improve the usefulness of this resource.

References

[1] S. El Kadiri and D. Kiritsis, 'Ontologies in the context of product lifecycle management: state of the art literature review', *Int. J. Prod. Res.*, vol. 53, no. 18, pp. 5657–5668, Sep. 2015, doi: 10.1080/00207543.2015.1052155.

[2] J. M. Park, J. H. Nam, Q. P. Hu, and H. W. Suh, 'Product Ontology Construction from Engineering Documents', in *2008 International Conference on Smart Manufacturing Application*, Goyang-si, South Korea: IEEE, Apr. 2008, pp. 305–310. doi: 10.1109/ICSMA.2008.4505663.

[3] K. I. Kotis, G. A. Vouros, and D. Spiliotopoulos, 'Ontology engineering methodologies for the evolution of living and reused ontologies: status, trends, findings and recommendations', *Knowl. Eng. Rev.*, vol. 35, p. e4, 2020, doi: 10.1017/S0269888920000065.

[4] M. Fernández-López, A. Gómez-Pérez, and N. Juristo, 'METHONTOLOGY: From Ontological Art Towards Ontological Engineering', in *AAAI Conference on Artificial Intelligence*, 1997. [Online]. Available: https://api.semanticscholar.org/CorpusID:10550105

[5] N. Noy, 'Ontology Development 101: A Guide to Creating Your First Ontology', 2001. [Online]. Available: https://api.semanticscholar.org/CorpusID:500106

[6] H. S. Pinto, S. Staab, C. Tempich, and Y. Sure, 'Distributed Engineering of Ontologies (DILIGENT)', in *Semantic Web and Peer-to-Peer*, S. Staab and H. Stuckenschmidt, Eds., Berlin/Heidelberg: Springer-Verlag, 2006, pp. 303–322. doi: 10.1007/3-540-28347-1_16.

[7] K. Kotis and G. A. Vouros, 'Human-centered ontology engineering: The HCOME methodology', *Knowl. Inf. Syst.*, vol. 10, no. 1, pp. 109–131, Jul. 2006, doi: 10.1007/s10115-005-0227-4.

[8] P. De Leenheer and C. Debruyne, 'DOGMA-MESS: A Tool for Fact-Oriented Collaborative Ontology Evolution', in *On the Move to Meaningful Internet Systems: OTM 2008 Workshops*, vol. 5333, R. Meersman, Z. Tari, and P. Herrero, Eds., in Lecture Notes in Computer Science, vol. 5333. , Berlin, Heidelberg: Springer Berlin Heidelberg, 2008, pp. 797–806. doi: 10.1007/978-3-540-88875-8_104.

[9] A. Garcia *et al.*, 'Developing Ontologies within Decentralised Settings', in *Semantic e-Science*, vol. 11, H. Chen, Y. Wang, and K.-H. Cheung, Eds., in Annals of Information Systems, vol. 11. , Boston, MA: Springer US, 2010, pp. 99–139. doi: 10.1007/978-1-4419-5908-9_4.

[10] F. Giunchiglia, B. Dutta, V. Maltese, and F. Farazi, 'A Facet-Based Methodology for the Construction of a Large-Scale Geospatial Ontology', *J. Data Semant.*, vol. 1, no. 1, pp. 57–73, May 2012, doi: 10.1007/s13740-012-0005-x.

[11] M. C. Suárez-Figueroa, A. Gómez-Pérez, and M. Fernández-López, 'The NeOn Methodology for Ontology Engineering', in *Ontology Engineering in a Networked World*, M. C. Suárez-Figueroa, A. Gómez-Pérez, E. Motta, and A. Gangemi, Eds., Berlin, Heidelberg: Springer Berlin Heidelberg, 2012, pp. 9–34. doi: 10.1007/978-3-642-24794-1_2.

[12] C. Debruyne, T.-K. Tran, and R. Meersman, 'Grounding Ontologies with Social Processes and Natural Language', *J. Data Semant.*, vol. 2, no. 2–3, pp. 89–118, Jun. 2013, doi: 10.1007/s13740-013-0023-3.

[13] A. Gangemi, ‘Ontology Design Patterns for Semantic Web Content’, in *The Semantic Web – ISWC 2005*, vol. 3729, Y. Gil, E. Motta, V. R. Benjamins, and M. A. Musen, Eds., in Lecture Notes in Computer Science, vol. 3729. , Berlin, Heidelberg: Springer Berlin Heidelberg, 2005, pp. 262–276. doi: 10.1007/11574620_21.
[14] E. Blomqvist, K. Hammar, and V. Presutti, ‘Engineering Ontologies with Patterns – The eXtreme Design Methodology’, in *Ontology Engineering with Ontology Design Patterns*, vol. 25: Ontology Engineering with Ontology Design Patterns, in Studies on the Semantic Web, vol. 25: Ontology Engineering with Ontology Design Patterns. , IOS Press, pp. 23–50.
[15] M. Vigo, S. Bail, C. Jay, and R. Stevens, ‘Overcoming the pitfalls of ontology authoring: Strategies and implications for tool design’, *Int. J. Hum.-Comput. Stud.*, vol. 72, no. 12, pp. 835–845, Dec. 2014, doi: 10.1016/j.ijhcs.2014.07.005.
[16] D. Spoladore and E. Pessot, ‘An evaluation of agile Ontology Engineering Methodologies for the digital transformation of companies’, *Comput. Ind.*, vol. 140, p. 103690, Sep. 2022, doi: 10.1016/j.compind.2022.103690.
[17] A. De Nicola and M. Missikoff, ‘A lightweight methodology for rapid ontology engineering’, *Commun. ACM*, vol. 59, no. 3, pp. 79–86, Feb. 2016, doi: 10.1145/2818359.
[18] S. Auer and H. Herre, ‘RapidOWL — An Agile Knowledge Engineering Methodology’, in *Perspectives of Systems Informatics*, vol. 4378, I. Virbitskaite and A. Voronkov, Eds., in Lecture Notes in Computer Science, vol. 4378. , Berlin, Heidelberg: Springer Berlin Heidelberg, 2007, pp. 424–430. doi: 10.1007/978-3-540-70881-0_36.
[19] A. Fader, S. Soderland, and O. Etzioni, ‘Identifying relations for open information extraction’, in *Proceedings of the Conference on Empirical Methods in Natural Language Processing*, in EMNLP ’11. USA: Association for Computational Linguistics, 2011, pp. 1535–1545.
[20] M. Banko, M. J. Cafarella, S. Soderland, M. Broadhead, and O. Etzioni, ‘Open information extraction from the web’, in *Proceedings of the 20th International Joint Conference on Artifical Intelligence*, in IJCAI’07. San Francisco, CA, USA: Morgan Kaufmann Publishers Inc., 2007, pp. 2670–2676.
[21] F. Wu and D. S. Weld, ‘Open Information Extraction Using Wikipedia’, in *Proceedings of the 48th Annual Meeting of the Association for Computational Linguistics*, J. Hajič, S. Carberry, S. Clark, and J. Nivre, Eds., Uppsala, Sweden: Association for Computational Linguistics, Jul. 2010, pp. 118–127. [Online]. Available: https://aclanthology.org/P10-1013
[22] Mausam, M. Schmitz, S. Soderland, R. Bart, and O. Etzioni, ‘Open Language Learning for Information Extraction’, in *Proceedings of the 2012 Joint Conference on Empirical Methods in Natural Language Processing and Computational Natural Language Learning*, J. Tsujii, J. Henderson, and M. Paşca, Eds., Jeju Island, Korea: Association for Computational Linguistics, Jul. 2012, pp. 523–534. [Online]. Available: https://aclanthology.org/D12-1048
[23] G. Angeli, M. Johnson, and C. D. Manning, ‘Leveraging Linguistic Structure For Open Domain Information Extraction’, in *Annual Meeting of the Association for Computational Linguistics*, 2015. [Online]. Available: https://api.semanticscholar.org/CorpusID:6015236
[24] L. Del Corro and R. Gemulla, ‘ClausIE: clause-based open information extraction’, in *Proceedings of the 22nd International Conference on World Wide Web*, in WWW ’13. New York, NY, USA: Association for Computing Machinery, 2013, pp. 355–366. doi: 10.1145/2488388.2488420.
[25] G. Angeli *et al.*, ‘Bootstrapped Self Training for Knowledge Base Population’, *Theory Appl. Categ.*, 2015, [Online]. Available: https://api.semanticscholar.org/CorpusID:18991323
[26] J. Chen, S. Xiao, P. Zhang, K. Luo, D. Lian, and Z. Liu, ‘BGE M3-Embedding: Multi-Lingual, Multi-Functionality, Multi-Granularity Text Embeddings Through Self-Knowledge Distillation’, 2024, doi: 10.48550/ARXIV.2402.03216.
[27] D. Dai, X. Xiao, Y. Lyu, S. Dou, Q. She, and H. Wang, ‘Joint Extraction of Entities and

Overlapping Relations Using Position-Attentive Sequence Labeling', *Proc. AAAI Conf. Artif. Intell.*, vol. 33, no. 01, pp. 6300–6308, Jul. 2019, doi: 10.1609/aaai.v33i01.33016300.

[28] Y. Lu *et al.*, 'Unified Structure Generation for Universal Information Extraction', in *Proceedings of the 60th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, S. Muresan, P. Nakov, and A. Villavicencio, Eds., Dublin, Ireland: Association for Computational Linguistics, May 2022, pp. 5755–5772. doi: 10.18653/v1/2022.acl-long.395.

[29] C. Wang, X. Liu, Z. Chen, H. Hong, J. Tang, and D. Song, 'DeepStruct: Pretraining of Language Models for Structure Prediction', in *Findings of the Association for Computational Linguistics: ACL 2022*, Dublin, Ireland: Association for Computational Linguistics, 2022, pp. 803–823. doi: 10.18653/v1/2022.findings-acl.67.

[30] P. Lewis *et al.*, 'Retrieval-augmented generation for knowledge-intensive NLP tasks', in *Proceedings of the 34th International Conference on Neural Information Processing Systems*, in NIPS'20. Red Hook, NY, USA: Curran Associates Inc., 2020.

[31] W. Shi *et al.*, 'REPLUG: Retrieval-Augmented Black-Box Language Models', 2023, doi: 10.48550/ARXIV.2301.12652.

[32] R. Falco, A. Gangemi, S. Peroni, D. Shotton, and F. Vitali, 'Modelling OWL Ontologies with Graffoo', in *The Semantic Web: ESWC 2014 Satellite Events*, vol. 8798, V. Presutti, E. Blomqvist, R. Troncy, H. Sack, I. Papadakis, and A. Tordai, Eds., in Lecture Notes in Computer Science, vol. 8798. , Cham: Springer International Publishing, 2014, pp. 320–325. doi: 10.1007/978-3-319-11955-7_42.

[33] A. Gangemi, N. Guarino, C. Masolo, A. Oltramari, and L. Schneider, 'Sweetening Ontologies with DOLCE', in *Knowledge Engineering and Knowledge Management: Ontologies and the Semantic Web*, vol. 2473, A. Gómez-Pérez and V. R. Benjamins, Eds., in Lecture Notes in Computer Science, vol. 2473. , Berlin, Heidelberg: Springer Berlin Heidelberg, 2002, pp. 166–181. doi: 10.1007/3-540-45810-7_18.

[34] N. Lazzari, S. De Giorgis, A. Gangemi, and V. Presutti, 'Sandra -- A Neuro-Symbolic Reasoner Based On Descriptions And Situations', 2024, doi: 10.48550/ARXIV.2402.00591.

[35] L. Grossman Liu *et al.*, 'A deep database of medical abbreviations and acronyms for natural language processing', *Sci. Data*, vol. 8, no. 1, p. 149, Jun. 2021, doi: 10.1038/s41597-021-00929-4.

[36] H. Gurulingappa, A. M. Rajput, A. Roberts, J. Fluck, M. Hofmann-Apitius, and L. Toldo, 'Development of a benchmark corpus to support the automatic extraction of drug-related adverse effects from medical case reports', *J. Biomed. Inform.*, vol. 45, no. 5, pp. 885–892, Oct. 2012, doi: 10.1016/j.jbi.2012.04.008.

[37] X. L. Li and P. Liang, 'Prefix-Tuning: Optimizing Continuous Prompts for Generation', in *Proceedings of the 59th Annual Meeting of the Association for Computational Linguistics and the 11th International Joint Conference on Natural Language Processing (Volume 1: Long Papers)*, C. Zong, F. Xia, W. Li, and R. Navigli, Eds., Online: Association for Computational Linguistics, Aug. 2021, pp. 4582–4597. doi: 10.18653/v1/2021.acl-long.353.