HACID - Deliverable

Domain knowledge graph instantiation and evaluation

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:	D2.2
Due date:	31.01.2025
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
V1	30/10/2024	Creation of draft document	Andrea Giovanni Nuzzolese	CNR
V2	17/11/2024	The outline and introduction of Bottom-up KG generation	Weilai Xu	CNR
V3	25/11/2024	Completion of the section Data Sources for Medical Diagnostics	Fiorela Ciroku	CNR
V4	28/11/2024	Completion of the first draft of Bottom-up KG generation with Entity-type experiment	Weilai Xu	CNR
V5	03/12/2024	Description of data sources and general editing	Andrea Giovanni Nuzzolese	CNR
V6	05/12/2024	Introduction and conclusion	Andrea Giovanni Nuzzolese	CNR
V7	05/12/2024	Functional, structural, and logical evaluation of the KGs	Alessandro Russo	CNR
V8	06/12/2024	Completion of Common Knowledge Completion of Medical Diagnostics: Ontology Network & Data Contribution to Climate Service: Ontology Network	Fiorela Ciroku, Miguel Ceriani	CNR
V9	10/12/2024	Restructuring of the Section describing the data generated for Climate Services	Andrea Giovanni Nuzzolese	CNR
V10	15/01/2025	Finalisation of the Entity Linking section	Octavianus Sinaga, Andrea Giovanni Nuzzolese	CNR
V11	18/01/2025	Revision and update of the bottom-up knowledge generation section	Vito Trianni, Weilai Xu	CNR
V12	28/01/2025	Revision of the manuscript	Gioele Barabucci	HDX
V13	30/01/2025	Final overall revision	Vito Trianni, Weilai Xu, Andrea Giovani Nuzzolese, Octavianus Sinaga, Alessandro Russo, Fiorela Ciroku, Miguel Ceriani	CNR

Table of content

1. Introduction

The development of domain-specific knowledge graphs is essential for enabling advanced data integration and semantic reasoning in open-ended tasks based on hybrid collective intelligence. Within the HACID project, this deliverable builds on the foundational methodologies outlined in **Deliverable 2.1** [1], which explored top-down and bottom-up approaches to domain knowledge engineering. Deliverable 2.1 focused on establishing the theoretical and methodological underpinnings for constructing knowledge graphs, including the use of eXtreme Design (XD) [2] and ontology design patterns [3] to ensure modularity, consistency, and reuse across domains. This prior work provided the conceptual grounding and core ontology modules that have been extended and finalised in this deliverable.

Here, we present the finalisation, instantiation, and evaluation of two distinct domain knowledge graphs tailored to **medical diagnostics** and **climate services**. We thus focus on the activities and results related to:

- **KGs finalisation:** the completion of the ontologies and data production processes (also in relation to the work presented in D2.1), ensuring that the conceptual models and data structures are fully defined and ready for use.
- **KGs instantiation:** The actual production of data and its availability in a triplestore, making the knowledge graphs accessible for querying and analysis.
- **KGs evaluation:** The qualitative and quantitative assessments of the different perspectives and approaches used in constructing and utilizing the knowledge graphs.

The medical diagnostics knowledge graph leverages structured data from SNOMED and Wikidata and mappings to established medical ontologies, such as UMLS, enhancing interoperability and providing a robust representation of medical knowledge. Meanwhile, the climate services knowledge graph consolidates diverse datasets, including CMIP and CORDEX outputs, supported by controlled vocabularies to represent climate variables, phenomena, and workflows essential for adaptation strategies and policy-making.

This deliverable details the data sources utilised, the ontology networks designed, and the methodologies employed for generating RDF data from the selected sources. It also provides an evaluation of these knowledge graphs, demonstrating their functional, logical, and structural robustness, while highlighting their utility for advancing insights in medical diagnostics and climate services.

2. Top-down knowledge generation

In this Section we describe the work done to design, implement, and deploy the knowledge graphs for medical diagnostics and climate services following a top-down approach. A top-down approach to knowledge graph (KG) design starts with high-level concepts, goals, and structures, refining them progressively into detailed implementations. This method prioritises coherence, semantic richness, and alignment with domain-specific goals, leveraging theoretical models, ontologies, and established standards. As outlined in the Deliverable 2.1 we applied the eXtreme Design (XD) methodology to enhance top-down modelling by introducing modularity, iterative refinement, and collaborative development. In next sections we describe the material used (cf. Section 2.1), the two KGs implemented with the associated ontology network (cf. Section 2.2), and the evaluation of such KGs (cf. Section 2.3).

2.1. Material

The Material section aims to describe data sources that have been selected, analysed and used for the extension of the knowledge graphs in the medical diagnostics and climate service domains.

2.1.1. Data Sources for Medical Diagnostics

As discussed in detail in D2.1, the domain KG for medical diagnostics is constructed with a robust foundation based on two primary information sources:

1. **SNOMED CT**, offering a standardized vocabulary that covers a vast array of medical terms, conditions, procedures, and concepts;
2. **HumanDX datasets**, consisting of a rich collection of clinical cases and their associated solutions, i.e., diagnoses proposed by medical professionals using the HumanDX platform.

Wikidata was then selected to expand the medical diagnostics knowledge graph. Wikidata serves as a comprehensive and collaborative knowledge base, offering structured data across various domains, including medicine. For the expansion of the existing KG, concepts such as disease, anatomical structure, substance, finding, procedure, and situation were selected. For instance, for the disease concept, Wikidata provides detailed information for the associated signs and symptoms, treatments, genetic associations, health care specialties, and therapies or drugs used for treatment. In addition, Wikidata associates concepts with medical identifiers such as SNOMED CT, ICD-9, ICD-10, UMLS codes, and others. By integrating these elements, Wikidata enables interoperability with other medical databases and more importantly, the existing knowledge graph. Furthermore, its open and collaborative framework ensures that the data is continuously updated and enriched by contributors worldwide, making it a valuable resource for both academic research and practical applications.
The procedure leading to the integration of the data from Wikidata into the knowledge graph by means of the mapping between UMLS, ICD-9, ICD-10 and SNOMED CT is described in detail in Section 2.2.3.2.

2.1.2. Data Sources for Climate Services

The knowledge graph for climate services is built upon a diverse array of data sources, each contributing specialized information. These data sources are described in the following.

- **CMOR Tables**: The Climate Model Output Rewriter (CMOR) tables are a foundational tool in climate modeling, designed to standardize the output of climate simulations for consistency and interoperability. Used in initiatives like the Coupled Model Intercomparison Project (CMIP), these tables define the metadata, variables, units, and dimensions required for compliance with data conventions such as

NetCDF Climate and Forecast (CF). CMOR Tables enable the harmonization of model outputs by specifying physical variables, temporal resolutions, and spatial aggregations, ensuring compatibility across diverse models and experiments. This standardization facilitates data sharing and multi-model comparisons on platforms like the Earth System Grid Federation (ESGF), playing a critical role in global climate assessments, including those by the IPCC. By ensuring uniformity and quality, CMOR Tables enhance the utility and reliability of climate data for research and decision-making. Sourced from GitHub, these include 1,273 Model Intercomparison Project (MIP) variables defined by masking and aggregation methods, 2,068 CMOR variables further specified by time granularity, and 90 units of measurement. This dataset is available in JSON format via Git.

- **Controlled Vocabularies (CVs)**: Controlled vocabularies (CVs) provide a standardized framework for describing key elements of climate model data, ensuring consistency and semantic interoperability across datasets. These vocabularies define essential metadata, including institution names, model identifiers, experiment labels, and variable attributes, which are critical for harmonizing outputs from different climate modeling centers and experiments. By enforcing uniform terminology and structure, CVs support the accurate organization, discovery, and integration of climate data within initiatives such as CMIP. The use of controlled vocabularies enhances the reliability of metadata-driven applications, such as knowledge graphs, and facilitates global collaborations in climate research and policy development. Also obtained from GitHub, this dataset encompasses 49 institutions and 134 models, providing key metadata in JSON format through Git access.
- **CMIP5 Datasets**: CMIP5 datasets, generated under the Fifth Coupled Model Intercomparison Project (CMIP5), represent a comprehensive collection of climate model outputs aimed at understanding past, present, and future climate changes. These datasets encompass simulations from multiple models, covering scenarios such as historical climate trends, future projections based on greenhouse gas concentration pathways, and idealized experiments. With over 6,000 datasets and 150+ simulations, CMIP5 provides standardized, high-resolution outputs across various spatial and temporal scales. Accessible via the Earth System Grid Federation (ESGF), the datasets conform to strict metadata and data format standards, enabling seamless integration and comparison. CMIP5 has been instrumental in advancing global climate assessments, including those by the Intergovernmental Panel on Climate Change (IPCC), supporting research on climate variability, impacts, and policy formulation. Accessed via the ESGF API, this dataset includes 152 simulations and a vast repository of 6,365 individual datasets, all delivered in JSON format.
- **CORDEX Domains**: CORDEX Domains represent a standardized framework for regional climate modeling within the Coordinated Regional Climate Downscaling Experiment (CORDEX). These domains define specific geographical regions where regional climate models are applied to downscale global climate projections, providing higher-resolution climate information for impact assessments and adaptation planning. Covering 14 global regions, CORDEX Domains ensure consistency in spatial boundaries and data formats, facilitating intercomparison and synthesis across models and studies. By focusing on regional climates, CORDEX enhances the understanding of localized climate impacts and variability, bridging the gap between global projections and actionable regional insights. These domains, managed via GitHub and available in CSV format, are essential for advancing climate services and supporting informed decision-making at regional levels. This

GitHub-hosted resource comprises 14 distinct domains, available in CSV format via Git.

- **CORDEX Datasets**: CORDEX datasets are a comprehensive collection of high-resolution regional climate projections generated through CORDEX. These datasets offer detailed climate information for the 14 defined CORDEX domains. Leveraging regional climate models, CORDEX datasets refine global climate model outputs to capture localized climate dynamics and impacts, enabling more accurate assessments of climate variability, extremes, and risks. CORDEX datasets are pivotal for climate research, regional impact studies, and the development of climate services, supporting adaptation strategies and policy decisions worldwide. Provided through the ESGF API, this dataset features 1,330 dynamical downscaling runs and 162,191 datasets, presented in JSON format.
- **Climdex**: Climdex is a collection of standardized climate indices designed to quantify changes in climate extremes using observational and modeled data. It includes 63 indices grouped into 9 categories, such as temperature and precipitation extremes, enabling consistent and robust analysis of climate variability and trends. Climdex resources are accessible via GitHub and the Climdex website, with datasets and documentation provided in Markdown and HTML formats. These indices are widely used in climate impact assessments, research, and adaptation planning, offering critical insights into the frequency, intensity, and duration of extreme climate events. Climdex plays a key role in bridging climate science and services by providing tools for translating raw climate data into actionable information for decision-making. The data is accessible in Markdown and HTML formats through Git and web interfaces.

Table 3 summarises the data sources used for generating the RDF data part of the KG for climate services by providing information about the source, numerosity, availability, and format.

Table 3. List of data sources for the climate services

Data	Source	Numerosity	How	Format
CMOR Tables	GitHub	● 1,273 MIP variables (physical variables specialised by masking/aggregation methods) ● 2,068 CMOR variables (further specialised by time granularity) ● 90 units of measure	Git	JSON
CVs	GitHub	● 49 institutions ● 134 models	Git	JSON
CMIP5 Datasets	ESGF (e.g., CEDA node)	● 152 simulations ● 6,365 datasets	API	JSON
CORDEX Domains	GitHub	● 14 domains	Git	CSV
CORDEX Datasets	ESGF (e.g., CEDA node)	● 1,330 dynamical downscaling runs ● 162,191 datasets	API	JSON
Climdex	GitHub, climdex.org	● 63 indices ● 9 categories	Git, Web	Markdown, HTML

2.2. Domain Knowledge Graphs

This section describes the advancement regarding the domain knowledge graphs both in general and for the two specific use cases.

2.2.1. Namespaces

The natural organisational choice is to have the two domain knowledge graphs as separate triple store instances, given that no need for cross-domain queries arose in the design process so far. In any case, cross-domain queries can be executed by federating SPARQL queries. At the same time, we would like URIs, in conformance with linked data principles, to get resolved via proper content negotiation to relevant HTML/RDF information for a resource. To facilitate such a mechanism the namespaces (specially the ones used for instances) are designed so that the initial part of the URL is sufficient to distinguish between the two knowledge graphs. For the ontologies the namespace used for the corresponding terms is always the ontology URI followed by a slash (/): for example, the climate services ontology is identified by `https://w3id.org/hacid/onto/ccso`, the namespace is `https://w3id.org/hacid/onto/ccso/`, and the class `ccso:Simulation` expands to `https://w3id.org/hacid/onto/ccso/Simulation`. Table 4 provides all the URIs for defining namespaces we introduced along with their usage and prefix.

Table 4. Ontology URIs, their intended usage, and namespace prefixes used for referencing them.

URI	Usage	NS Prefix
`https://w3id.org/hacid/onto/`	Namespace for all HACID ontology network	
`https://w3id.org/hacid/onto/common/`	Namespace for ontology modules used in both use cases	
`https://w3id.org/hacid/onto/common/top/`	Top-level ontology module	`top:`
`https://w3id.org/hacid/onto/common/agentrole/`	Agent Role ontology module	`ar:`
`https://w3id.org/hacid/onto/common/judgement/`	Judgement ontology module	`jdg:`
`https://w3id.org/hacid/onto/common/naming/`	Naming ontology module	`nm:`
`https://w3id.org/hacid/onto/common/evidence/`	Evidence Reporting ontology module	`ev:`
`https://w3id.org/hacid/onto/mdx/`	Medical diagnostics ontology module	`mdx:`
`https://w3id.org/hacid/onto/hsct/`	Snomed Clinical Terms ontology module	`hsct:`
`https://w3id.org/hacid/onto/ccso/`	Climate services ontology module	`ccso:`
`https://w3id.org/hacid/onto/data/`	Data description ontology module (variables and dimensional spaces)	`data:`
`https://w3id.org/hacid/data/mdx/`	Namespace for all instances in the medical domain knowledge graph	`mdxdata:`
`https://w3id.org/hacid/data/cs/`	Namespace for all instances in the climate services domain knowledge graph	`csdata:`
`https://w3id.org/hacid/data/cs/variable/`	Namespace for variables in the climate services domain knowledge graph	`variable:`
`https://w3id.org/hacid/data/`<use-case>`/`<type>`/`	General pattern for instances of a specific type in one of the two use cases. If needed,	

	the pattern can be extended with further levels in the path.

2.2.2. Common Knowledge

This section describes the update of the core and top level modules to fulfil the requirements from both the domain modules, medical diagnostics and climate services.

As reported in Deliverable 2.1 [1], the core and top level modules are designed following the eXtreme Design (XD) [2] ontology modelling methodology that is based in modularity and the reuse of ontology design patterns (ODP) [3]. Considering the modularity aspect of the XD methodology, the extension of the domain modules prompted the need to extend the core and top level modules as well. The extension of the core and top level modules is grounded in the reuse of patterns from the DOLCE UltraLite+DnS (DUL) ontology[3] [4].

With the last update (as of the time of this writing), the top level ontology module has been expanded to fully integrate the "workflow" and "sequence" ontology design patterns. This means that all conceptual structures, relationships, and constraints defined by these patterns have been integrated, ensuring comprehensive support for representing workflows and sequences within the ontology. Both patterns are used in a new module from the Climate Service domain, named Climate Service Workflow which is described in Section 2.2.4.1. Meanwhile, in Figure 1 is shown the Workflow pattern which aims to define roles, tasks and a specific structure for tasks to be executed. The pattern includes concepts such as *Plan*, *Workflow*, *Workflow execution*, *Action*, *Task*, *Role*, *Agent*. The workflow defines the roles and tasks that need to be performed, while the workflow execution represents a specific execution that satisfies the workflow. The workflow execution includes an action that executes a task and has as a participant an agent. In addition, the AgentRole concept couples the agent with the role that it holds.

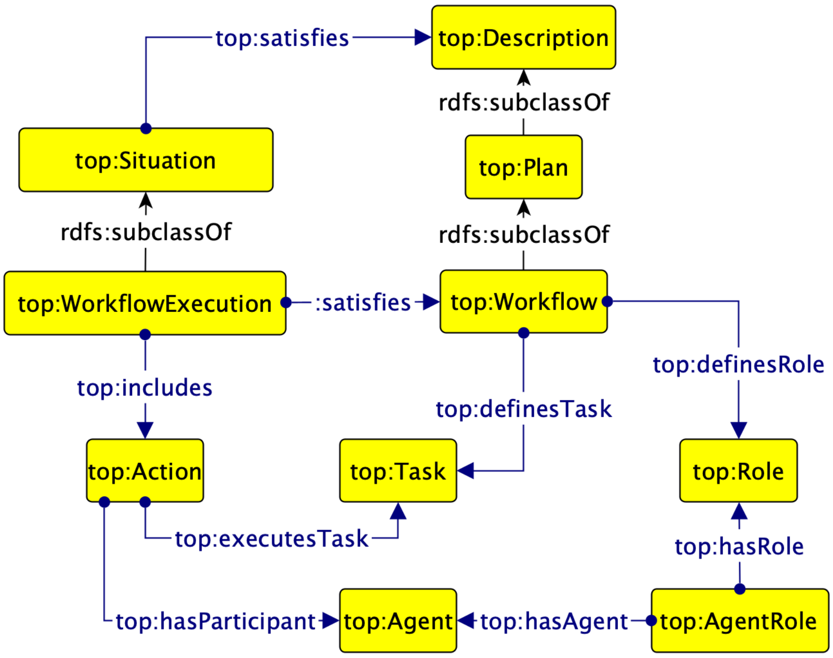


Figure 1. The Workflow pattern.

[3] http://www.ontologydesignpatterns.org/ont/dul/DUL.owl, last visited on December 6th 2024.

The sequence pattern (Figure 2) is used to express the order in which the tasks are executed. There are two main properties such as *follows* and *precedes*, and a respective subproperty named *directlyFollows* and *directlyPrecedes*. This specification of the order of tasks gives a clear overview on how the tasks are interconnected with each other. It is critical to express the ordering of the tasks since there might be cases when the workflow is split into two different flows.

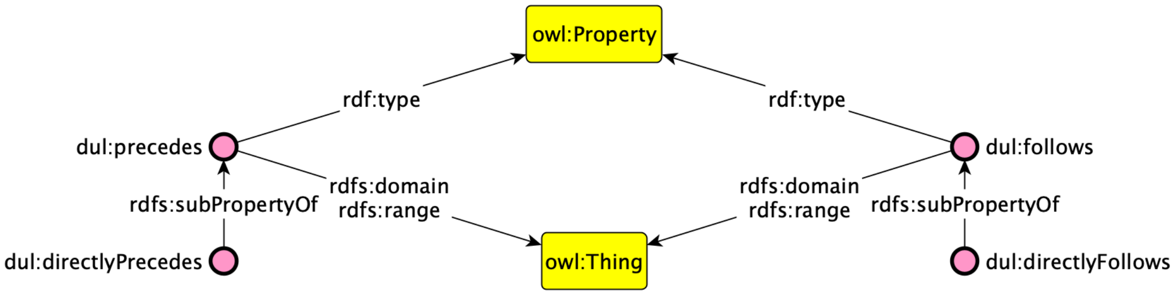


Figure 2. The Sequence pattern.

Table 5 reports the competency questions[4] that can be answered by the top module after the update with the aforementioned ontology design patterns.

Table 5. Competency questions for capturing knowledge on workflows and sequences. This table extends Table 1 in D2.1, where an initial set of 16 CQs for the top-level module were identified.

ID	Competency question
top-17	Which task is included in a workflow?
top-18	How is a task sequenced in a workflow?
top-19	Which role is associated with a task?
top-20	What role does an agent have in the execution of a task?
top-21	Which action executes a task?
top-22	Which task can be performed by different roles?
top-23	Which workflow execution satisfies a workflow?

2.2.3. Medical Diagnostics

This section details the enhancement of the Medical Diagnostics knowledge graph using data extracted from Wikidata. The integration process focused on leveraging Wikidata's structured data, including medical identifiers, relationships, and domain-specific properties. By incorporating this information, the extended knowledge graph offers improved connectivity and enriched insights for medical diagnostics.

[4] In ontology design, *competency questions* are a set of questions that define the intended scope and capabilities of an ontology. They help specify the types of information the ontology should represent and the kinds of queries it should be able to answer, guiding its structure and development.

2.2.3.1. Ontology Network

Wikidata represents a robust and reliable resource for the development and augmentation of knowledge graphs due to its structured, open, and semantically rich nature. The concepts extracted from Wikidata are instances of or subclasses of the concepts *disease*, *anatomical structure*, *substance*, *finding*, *procedure*, and *situation.* The main criteria for the selection of the data is that the concepts have at least one of the following medical IDs: SNOMED ID, UMLS, ICD-9, ICD-10. To retrieve these data, we made an API request to the Wikidata Query Service[5] (WQS) with the following query. The illustrative query selects the subject, property and object of a triple together with their label where the subject of the triple is an instance of (P31) or a subclass of (P279) the concept Disease (Q12136). Furthermore, the subject must have at least one of the specified medical IDs, so it searches for the properties corresponding to SNOMED CT ID, ICD-9, ICD-10, and UMLS. Lastly, it calls the service for the labelling of the results in the English language.

```
SELECT ?item ?itemLabel ?property ?propertyLabel ?value ?valueLabel
WHERE {
  ?item wdt:P31/wdt:P279 wd:Q12136.
  ?item ?property ?value .
  VALUES ?property {
    wdt:P5806   # SNOMED CT ID
    wdt:P493    # ICD-9 ID
    wdt:P494    # ICD-10 ID
    wdt:P2892   # UMLS ID
    }
  SERVICE wikibase:label {
    bd:serviceParam wikibase:language "[AUTO_LANGUAGE],en".
  }
}
```

Before extracting further data, for each of the concepts, we ranked the properties from most to least occurring in a triple where the concept is the subject. Only a restricted number of properties were selected to be incorporated into the knowledge graph, chosen based on their ranking and perceived relevance to its development and purpose. For instance, for the concept Disease the most occurring and relevant properties are *subclass of, instance of*, *genetic association*, *symptoms and signs*, *health specialty*, *drug of therapy used for treatment*, etc, as shown in Figure 3. A considerable number of properties such as *has part*, *has effect*, *different from*, etc, is shared among the concepts, so the overall number of the selected properties is not overwhelming. Nonetheless, is it quite inclusive and descriptive of the concepts.

[5] https://query.wikidata.org/, last visited on December 6th 2024.

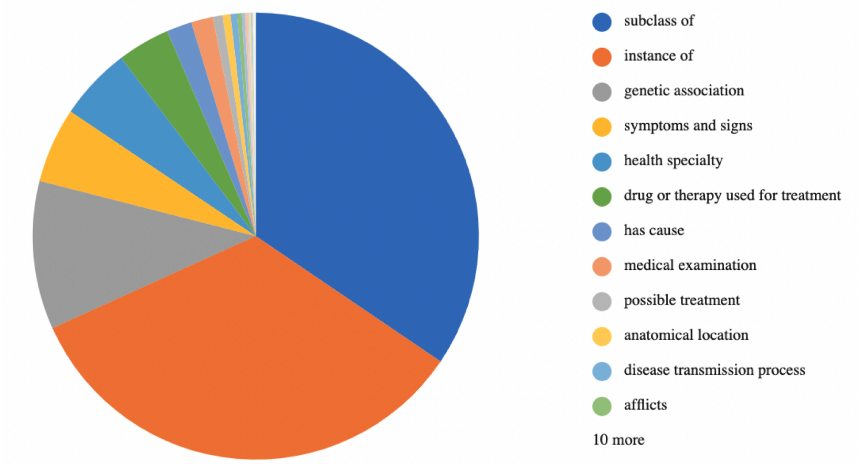


Figure 3. Occurrence of the properties in triples where an individual of Disease is the subject.

After the selection of the relevant properties to be added in the knowledge graph, the second data extraction process started. Again, the extraction consists of asking WQS to select triples where the subject is one of the concepts retrieved in the first query and the property is any of the list. The query returns the triples with the URIs and labels as shown in the query below.

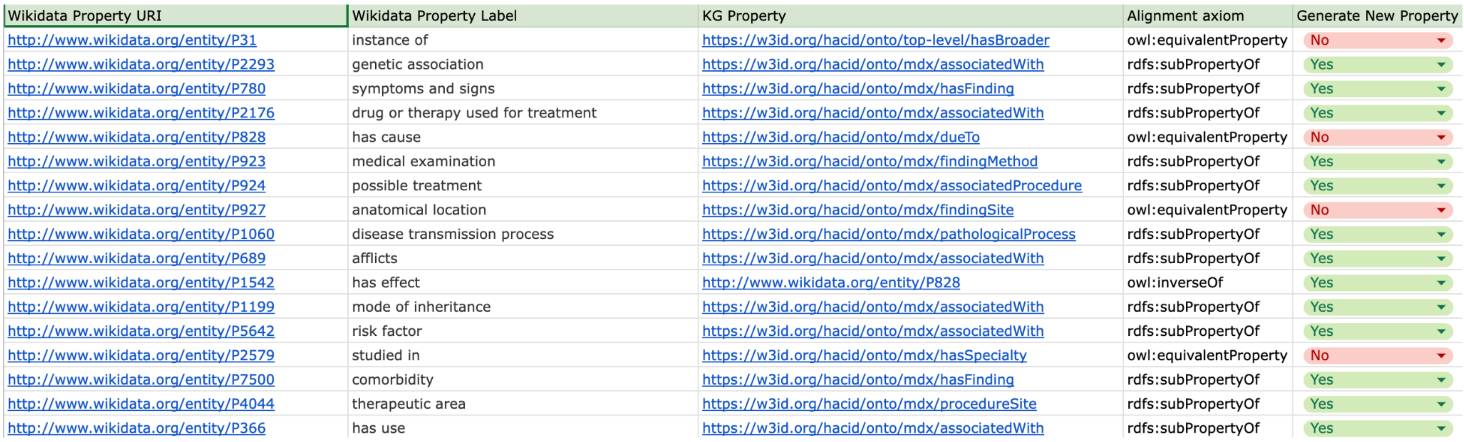
```
SELECT ?item ?itemLabel ?property ?propertyLabel ?value ?valueLabel
WHERE {
    OPTIONAL {
    wd:{item_id} ?property ?value .
    VALUES ?property {
      wdt:P2293   # Genetic association
      wdt:P780    # Symptoms and signs
      wdt:P1995   # Medical specialty
      wdt:P2176   # Drug used for treatment
      wdt:P828    # Has cause
      wdt:P923    # Medical examinations
      wdt:P924    # Possible treatment
      wdt:P927    # Anatomical location
      wdt:P1060   # Disease transmission process
      wdt:P689    # Afflicts
      wdt:P1542   # Has effect
      wdt:P527    # Has part(s)
      wdt:P460    # Said to be the same as
      wdt:P1199   # Mode of inheritance
      wdt:P5131   # Possible medical findings
      wdt:P361    # Part of
      wdt:P1552   # Has characteristic
      wdt:P5642   # Risk factor
      wdt:P2579   # Studied in
      wdt:P7500   # Comorbidity
    }
  }
  SERVICE wikibase:label {
    bd:serviceParam wikibase:language "[AUTO_LANGUAGE],en".
  }
}
```

Once the data is extracted from Wikidata, we analysed the properties and manually aligned them to the existing knowledge graph. The alignment consists of different relationships between the properties:

1. The properties are equivalent (e.g., Wikidata property instanceOf and the KG property hasBroader are equivalent to each other).

2. A property is a sub-property of another property (e.g., Wikidata property geneticAssociation is a sub-property of the KG property associatedWith).
3. A property is the inverse of another property (e.g., Wikidata property activeIngredientIn is inverse of KG property hasActiveIngredient).

Figure 4 shows the alignment of all the Wikidata properties described with the URI and label to the KG properties. The last column of the table indicated whether it is necessary to generate a new property for the integration of the data in the existing knowledge graph.



Wikidata Property URI	Wikidata Property Label	KG Property	Alignment axiom	Generate New Property
http://www.wikidata.org/entity/P31	instance of	https://w3id.org/hacid/onto/top-level/hasBroader	owl:equivalentProperty	No
http://www.wikidata.org/entity/P2293	genetic association	https://w3id.org/hacid/onto/mdx/associatedWith	rdfs:subPropertyOf	Yes
http://www.wikidata.org/entity/P780	symptoms and signs	https://w3id.org/hacid/onto/mdx/hasFinding	rdfs:subPropertyOf	Yes
http://www.wikidata.org/entity/P2176	drug or therapy used for treatment	https://w3id.org/hacid/onto/mdx/associatedWith	rdfs:subPropertyOf	Yes
http://www.wikidata.org/entity/P828	has cause	https://w3id.org/hacid/onto/mdx/dueTo	owl:equivalentProperty	No
http://www.wikidata.org/entity/P923	medical examination	https://w3id.org/hacid/onto/mdx/findingMethod	rdfs:subPropertyOf	Yes
http://www.wikidata.org/entity/P924	possible treatment	https://w3id.org/hacid/onto/mdx/associatedProcedure	rdfs:subPropertyOf	Yes
http://www.wikidata.org/entity/P927	anatomical location	https://w3id.org/hacid/onto/mdx/findingSite	owl:equivalentProperty	No
http://www.wikidata.org/entity/P1060	disease transmission process	https://w3id.org/hacid/onto/mdx/pathologicalProcess	rdfs:subPropertyOf	Yes
http://www.wikidata.org/entity/P689	afflicts	https://w3id.org/hacid/onto/mdx/associatedWith	rdfs:subPropertyOf	Yes
http://www.wikidata.org/entity/P1542	has effect	http://www.wikidata.org/entity/P828	owl:inverseOf	Yes
http://www.wikidata.org/entity/P1199	mode of inheritance	https://w3id.org/hacid/onto/mdx/associatedWith	rdfs:subPropertyOf	Yes
http://www.wikidata.org/entity/P5642	risk factor	https://w3id.org/hacid/onto/mdx/associatedWith	rdfs:subPropertyOf	Yes
http://www.wikidata.org/entity/P2579	studied in	https://w3id.org/hacid/onto/mdx/hasSpecialty	owl:equivalentProperty	No
http://www.wikidata.org/entity/P7500	comorbidity	https://w3id.org/hacid/onto/mdx/hasFinding	rdfs:subPropertyOf	Yes
http://www.wikidata.org/entity/P4044	therapeutic area	https://w3id.org/hacid/onto/mdx/procedureSite	rdfs:subPropertyOf	Yes
http://www.wikidata.org/entity/P366	has use	https://w3id.org/hacid/onto/mdx/associatedWith	rdfs:subPropertyOf	Yes

Figure 4. Alignment of all the Wikidata properties described with the URI and label to the KG properties.

Effectively, at the ontological level the knowledge graph is enhanced with additional properties, offering a more detailed representation of the domain, along with new identifiers that establish connections to other medical terminologies.

2.2.3.2. Data

At the computational level, there are several challenges to be taken into consideration regarding the extraction of the data from Wikidata and the generation of the medical ID mappings from UMLS, ICD-9, and ICD-10 to SNOMED CT. These include *data heterogeneity*, as different sources use varying formats and levels of granularity; *data extraction complexity*, as querying Wikidata efficiently and within constraints on query execution requires pagination and other techniques; *inconsistencies and gaps*, where missing or conflicting mappings must be resolved; *scalability*, due to the large volume of medical terms and relationships; and *semantic alignment*, ensuring that concepts from different coding systems are correctly matched to maintain accuracy and interoperability.

As mentioned in the above section, the extraction of the data is realised by making API calls to the Wikidata Query Service. The first encountered issue was that the query takes too much time due to its complexity and the amount of data that it selects and filters. Therefore, it was necessary to split the query into two, one selecting concepts with medical IDs and the other selecting triples with specific properties, and to add waiting time between each call to avoid getting timeout errors. The execution of the first query resulted in the selection of 24,122 unique concepts and the extraction of 38,735 triples from Wikidata, accounting for the fact that many concepts are associated with multiple IDs. As shown in Figure 5, instances or subclasses of the concepts substance, disease, and anatomical structure constitute 55.6%, 29.4%, and 12.2% of the total triples, respectively. In contrast, the categories medical finding, procedure, and situation account for only 1.2%, 1.1%, and 0.5% of the overall triples.

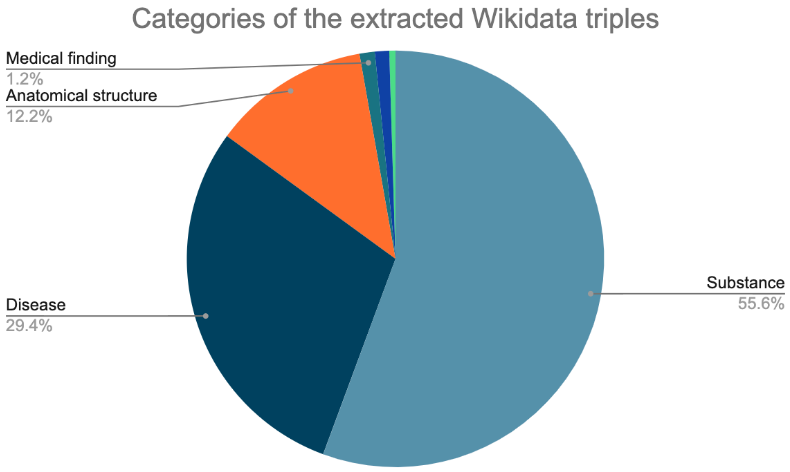


Figure 5. Categories of the extracted Wikidata triples.

The second query, which focused on extracting triples with the selected properties, yielded a total of 112,000 triples. As anticipated, the distribution of the retrieved triples aligns with the proportions of the concept categories. Specifically, as shown in Figure 6, 50% of the triples have the concept **Substance** as their subject, 35% are associated with the concept **Disease**, and 11.8% pertain to the concept **Anatomical Structure**. Triples from other smaller categories account for just 2.3% of the total number.

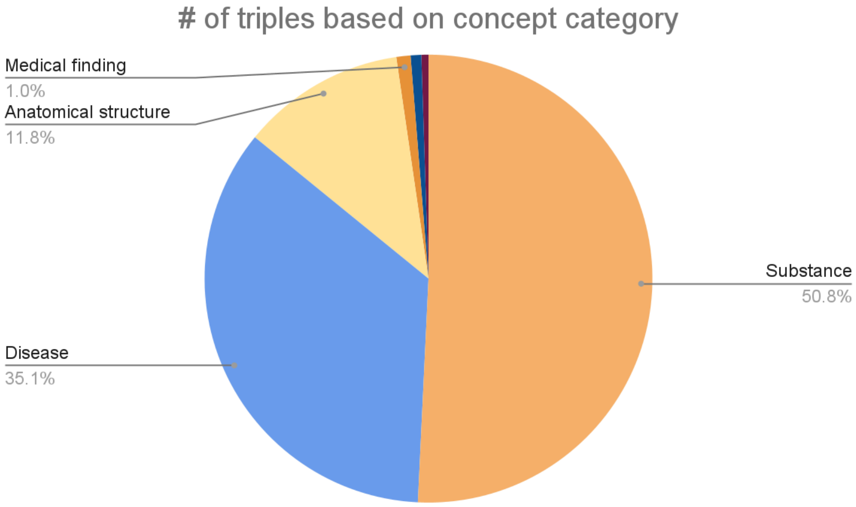


Figure 6. Number of triples based on concept category.

The example below shows one of the results from the query that retrieves the medical IDs. In this case, the subject is `otosalpingitis`, an instance of the concept disease, and it is identified by the UMLS ID `C0155428`. The same JSON structure is present in the results of the second query. The scripts for extracting the data from Wikidata and the results can be found in GitHub[6].

[6] https://github.com/hacid-project/knowledge-graph/tree/main/Wikidata%20Extraction, last visited on December 6th 2024.

```
...
{
        "item": {
          "type": "uri",
          "value": "http://www.wikidata.org/entity/Q4173417"
        },
        "property": {
          "type": "uri",
          "value": "http://www.wikidata.org/prop/direct/P2892"
        },
        "value": {
          "type": "literal",
          "value": "C0155428"
        },
        "itemLabel": {
          "xml:lang": "en",
          "type": "literal",
          "value": "otosalpingitis"
        },
        "propertyLabel": {
          "type": "literal",
          "value": "UMLS CUI"
        },
        "valueLabel": {
          "type": "literal",
          "value": "C0155428"
        }
...
```

As per the generation of the medical ID mappings, once the data was extracted from Wikidata, it was noticed that there are different categories of identified concepts such as:

1. Concepts that are identified by only one medical ID,
2. Concepts that are identified by multiple IDs from different medical classifications (i.e. a UMLS ID and an ICD-10 ID), and
3. Concepts that are identified by multiple IDs from the same medical classification (i.e. multiple UMLS IDs).

For all the categories, if the SNOMED CT ID exists, there is no mapping needed. Otherwise, the mapping from UMLS, ICD-9, or ICD-10 ID is performed. We need to emphasise that even though the mappings were computed by aiming for an exact match, there are multiple cases when an identifier is mapped to multiple SNOMED IDs. This happens for two main reasons: 1) there are different levels of specifications between the terminologies, and 2) the mapping itself might link to different versions of SNOMED ID. During the mapping, we selected only IDs that are currently active, to avoid mappings that direct to deprecated versions of SNOMED CT. The mapping between SNOMED CT and UMLS is realised by using a package named “UMLS to SNOMED CT and ICD-10 Mapper”[7] available in GitHub and developed by the NLPie Research team. The package enables the mapping of UMLS IDs to SNOMED CT and ICD-10 IDs by leveraging the UMLS Metathesaurus. It generates JSON files that represent the mappings, using one of three methods: exact matches

[7] https://github.com/nlpie-research/umls-to-snomed-icd10-mapper/tree/main, last visited on December 6th 2024.

(EXACT), broader relationships (RO), or parent-child hierarchical connections (PAR_CHD). For each approach, the script creates two JSON files: one containing SNOMED CT mappings and the other containing ICD-10 mappings. The JSON files have the following structure:

```
{
  "C0000039": [
    "102735002"
  ],
  "C0000163": [
    "112116001",
    "83489009",
    "40181004"
  ]
}
```

Meanwhile, the mapping of SNOMED CT IDs to ICD-9 and ICD-10 IDs are released periodically with the release of the SNOMED CT ontology. The text file containing the mapping of SNOMED CT to ICD-9 includes information such as the ICD code and name, the indication whether the ICD code is currently active or in use, inpatient usage, SNOMED ID and SNOMED fully specified name. Table 6 shows an example of such a mapping between ICD and SNOMED CT.

Table 6: Example of alignment between ICD concepts and SNOMED Clinical Terms. Each ICD concept is reported with its code, name, and information about validity in the current release. Instead, SNOMED CT are associated with their identifier and fully specified name.

ICD CODE	ICD NAME	IS CURRENT ICD	IP USAGE	SNOMED CID	SNOMED FSN
99.04	Transfusion of packed cells	1	6.574725	288170000	Packed blood cell transfusion (procedure)

The text file containing the mapping of SNOMED CT to ICD-10 is more thorough in terms of information that it provides. Each record has a unique identifier and a time when the mapping became active. It also indicates if the mapping is active, the module within SNOMED CT to which this mapping belongs, reference set ID, the SNOMED CT concept ID being mapped, the group of mappings within a reference set for cases where one SNOMED concept maps to multiple targets, the priority of the mapping within a group, the rule or condition for the mapping to apply, additional advice or comments about the mapping, the target code in the mapped system, the correlation or relationship between the source SNOMED CT concept and the target code, and the category of the mapping. Table 7 shows an example of such a mapping.

Table 7: Example of alignment between ICD-10 concepts and SNOMED Clinical Terms. Each mapping record has a unique identifier and it specifies if it is an active mapping with the timestamp of when it became active. The record includes the SNOMED CT concept that is being mapped with the ICD-10 ID, together with information such as the module within SNOMED CT to which the mapping belongs, the reference set ID, mapping group, priority, rule, advice, category ID and correlation ID.

ID	Module ID	RefSet ID	Component ID	Group	Target	Correlation ID	Category ID
xyz	449080006	447562003	254153009	1	Q79.8	447561005	447637006

In the data sample below, it is shown the mapping of the medical IDs for the Wikidata concept *Q182155*, corresponding to the infectious disease *herpes zoster*. As seen, this concept is identified by IDs from UMLS, ICD-9, and ICD-10. From these IDs, we were able to map only the UMLS ID to multiple SNOMED IDs. The outcome documents the type of mapping conducted, along with the source and target IDs. The complete script for the mapping is available on GitHub[8].

```
...
"http://www.wikidata.org/entity/Q182155": {
        "Original_IDs": [
            {
                "Type": "http://www.wikidata.org/prop/direct/P2892",
                "Value": "C0019360"
            },
            {
                "Type": "http://www.wikidata.org/prop/direct/P494",
                "Value": "B02"
            },
            {
                "Type": "http://www.wikidata.org/prop/direct/P493",
                "Value": "053"
            }
        ],
        "Mapped_SNOMED_IDs": [
            {
                "Type": "UMLS-to-SNOMED",
                "Source_ID": "C0019360",
                "SNOMED_IDs": [
                    "4740000",
                    "186533008",
                    "186514003",
                    "154326002"
                ]
            }
        ]
    }
...
```

[8] https://github.com/hacid-project/knowledge-graph/blob/main/Wikidata%20Extraction/Scripts/MedicalID_Mapping.py, last visited on December 6th 2024.

The results presented in Table 8 summarise the processing and mapping of various concept categories to SNOMED IDs. Among the categories, **Disease** shows significant mapping activity, with 6,258 processed items leading to 6,548 completed mappings, although 2,036 items lacked a corresponding SNOMED ID, resulting in 67% of items successfully mapped. **Substance**, the largest category with 12,583 processed items, achieved 9,891 mappings, but a substantial 6,712 items were not mapped, yielding a mapping percentage of 46%. **Anatomical Structure** processed 4,550 items, completing 1,003 mappings while 3,604 items remained unmapped, resulting in only 20% mapped. Smaller categories such as **Medical Finding**, **Procedure**, and **Situation** showed higher mapping efficiency relative to their size. For instance, **Medical Finding** had 261 processed items, achieving 320 mappings with 82% mapped, while **Procedure** processed 369 items with 73% mapped. Finally, **Situation** had 101 processed items, achieving a 52% mapping success rate. These results highlight the variability in mapping success across categories, influenced by the availability and alignment of SNOMED IDs with the processed concepts. The primary emphasis of the mapping process is on accuracy rather than the quantity of mapped concepts. To ensure precision, we prioritized achieving exact matches between IDs. It is possible that employing alternative methods, such as relative relationships or parent-child hierarchies, could increase the number of mapped items. However, these approaches do not provide the same assurance of mapping quality.

Table 8: The mapping between various concept categories and SNOMED IDs.

Concept category	Processed items	Total mappings completed	Items without SNOMED ID	% of mapped items to SNOMED
Disease	6258	6548	2036	67%
Substance	12583	9891	6712	46%
Anatomical structure	4550	1003	3604	20%
Medical finding	261	320	47	82%
Procedure	369	299	98	73%
Situation	101	92	48	52%

2.2.4. Climate Services

This section describes the domain knowledge graph built to support climate services. While the focus is on the instantiation, the ontology was also updated and it is thus described, focusing on novel aspects.

2.2.4.1. Ontology Network

In respect to the ontology network presented in D2.1 [1], there have been some modifications based on novel/refined requirements both elicited from continuous interaction with domain experts and arising from the need to adequately represent a set of relevant data sources (see 3.3.2). The main change to the previous version is the addition of a module allowing a richer description of the climate service handling process, which has been previously described in a quite abstract way for the lack of more detailed information. Also the description of variables and datasets was updated with the aim of fitting the complex ecosystem of variables and indices of different kinds in a coherent framework. Finally, the climate services ontology module is now called Core Climate Services Ontology[9] (CCSO).

Climate Projections: Climate Models and Simulations

The representation of climate models and simulations is mostly preserved from the previous version. Specifically, the taxonomy of models (cf. Figure 5) is the same and it is shown for completeness in the following diagram, which, in respect of the corresponding one in D2.1, clarifies the alignment with the top module of the ontology and the namespace used for the classes.

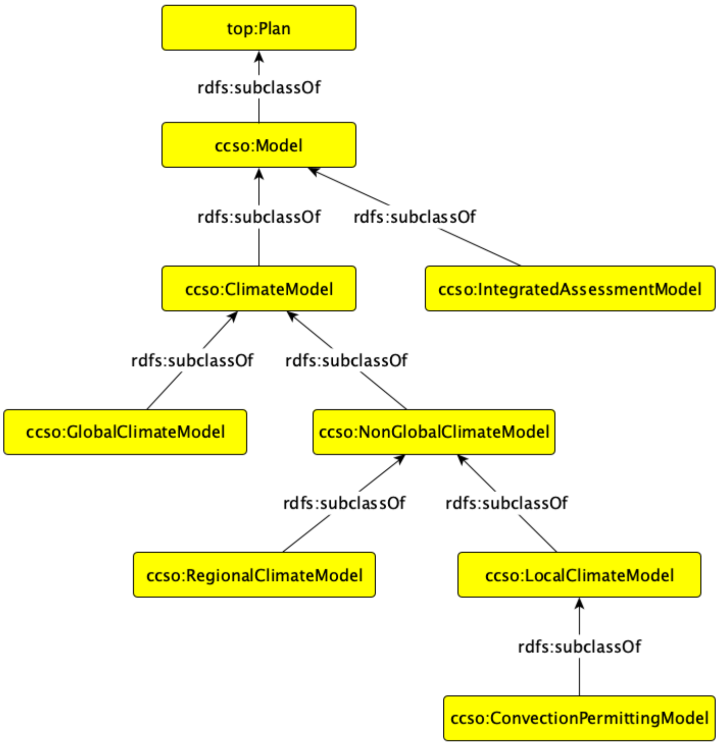


Figure 5. Taxonomy of Climate Models.

Regarding simulations, the representation is slightly simplified, excluding a specific formalisation of simulations with integrated assessment models which is outside of current requirements (no specific data sources or competency questions have arisen). Coherently with that choice and with domain usage of the term, an `ccso:EmissionScenario` is now not necessarily a quantitative spatiotemporal dataset of emissions and covers also SSPs and RCPs (instances of respectively `ccso:SharedSocioeconomicPathway` and `ccso:GreenhouseGasConcetrationPathway`), for which the general class Pathway was previously used. The property `ccso:refersToScenario` allows the direct association of a climate simulation with a considered emission scenario. Also in this case, alignment with the top ontology module is clarified. Figure 6 shows a detailed diagram of on Climate Simulations and Emission Scenarios.

[9] https://github.com/hacid-project/knowledge-graph/blob/main/ontologies/ccso/ccso.owl last visited on January 9th 2025.

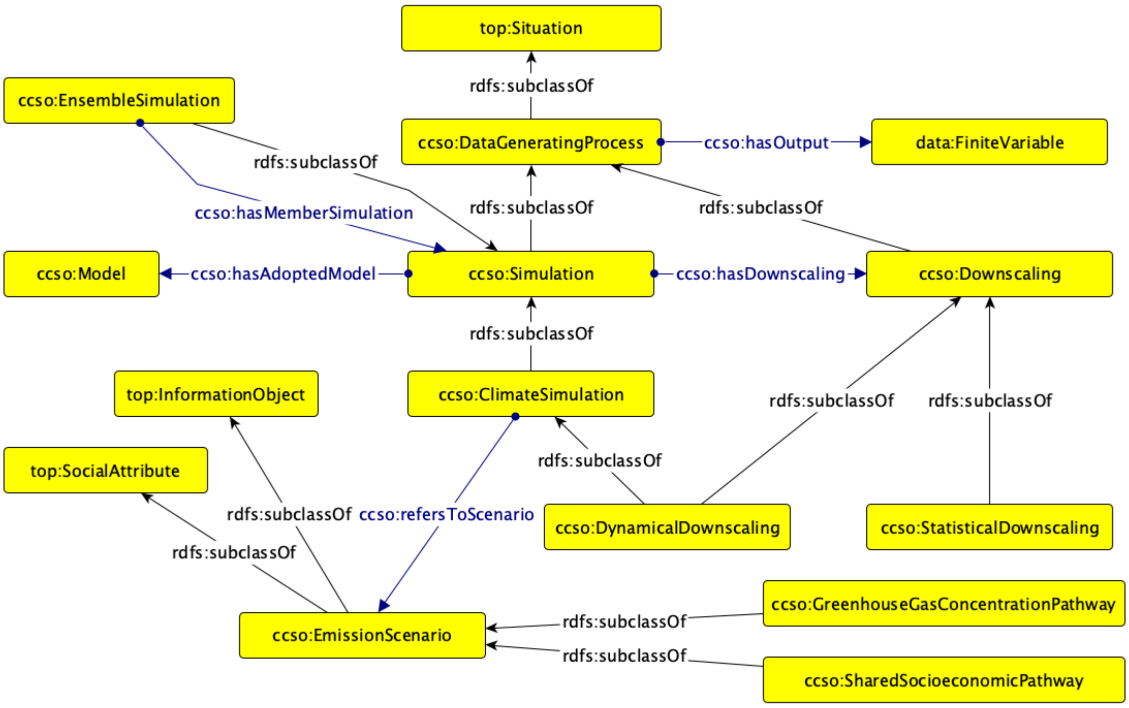


Figure 6. Diagram on Climate Simulations and Emission Scenarios.

Data Specification: Variables, Dimensional Spaces, and Datasets

Based on the requirements and previous analysis in D2.1, we elicited the need for a data-centric representation, based on the properties of datasets in terms of variables (which may be dependent from other variables or not, computationally derived from other variables or not) and the dimensional spaces they take values in (which can be discrete or continuous, made of points or extended regions, limited or not, derived-from/part-of other dimensional spaces). For that purpose a dedicated submodule of CCSO has been developed, associated with the namespace `https://w3id.org/hacid/onto/data`, here shortened with the prefix `ccso:`.

The aim is to represent the concept of variables in a flexible way, while preserving well-defined semantics. Here, as usual in mathematics and other fields, a variable is an abstraction that has meaning in the context of some relation between different variables and can be associated to different values in a set that is usually well defined (e.g., $x \in \mathbb{R}$). In the ontology module presented here, a `data:Variable` is akin to that concept and the set of possible values is represented by a data:DimensionalSpace; the two are linked via the `data:hasValuesOn` property. To a variable that is a `data:DependentVariable` can be implicitly associated with a function having as range the associated dimensional space and as domain the cartesian set of the dimensional spaces of the variables from which it depends. A `data:IndependentVariable`, conversely, is a variable that does not depend on other variables.

A variable can be *aggregating*, i.e. include aggregation mechanisms whose execution is deferred to derived variables. This allows to represent variables defined using one or more aggregation functions for combining data along one or more independent variables, without specifying the actual granularity. A general purpose example of usage is representing as variables the SQL expressions including aggregation functions (e.g. AVG(...)) which are then derived to actual serialisable variables by the usage of specific GROUP BY clauses. In the context of climate services, the CMOR Tables variables (e.g., tasmax) fall into this category. From the ontology point of view, a (dependent) variable is aggregating if it offers one or more aggregations: i.e. it is linked with one or more instances of data:Aggregation via the property data:definesAggregation. Each aggregation is associated with one variable (which must be a variable the dependent variable depends on) and optionally one or more suggested options for the adopted aggregation grid (i.e. specific granularity).

A data:FiniteVariable is a variable for which the set of all the possible values is finite. For an independent variable it means being defined on a finite dimensional space (instance of data:FiniteDimensionalSpace); for a dependent variable it means that all the variables it depends on are finite variables. The concept of data:FiniteVariable is important because it corresponds to variables that can be fully represented extensively with a finite amount of resources. It corresponds functionally to the concept of a dataset. The property data:hasDataSerialisation links such a dataset with a data source (data:DataSource) holding one or more concrete serialisations of it. A data:DataSource (e.g., an URL resolving to a dump of a dataset) is associated to one or more supported data formats (data:DataFormat) via the data:hasAvailableDataFormat property.

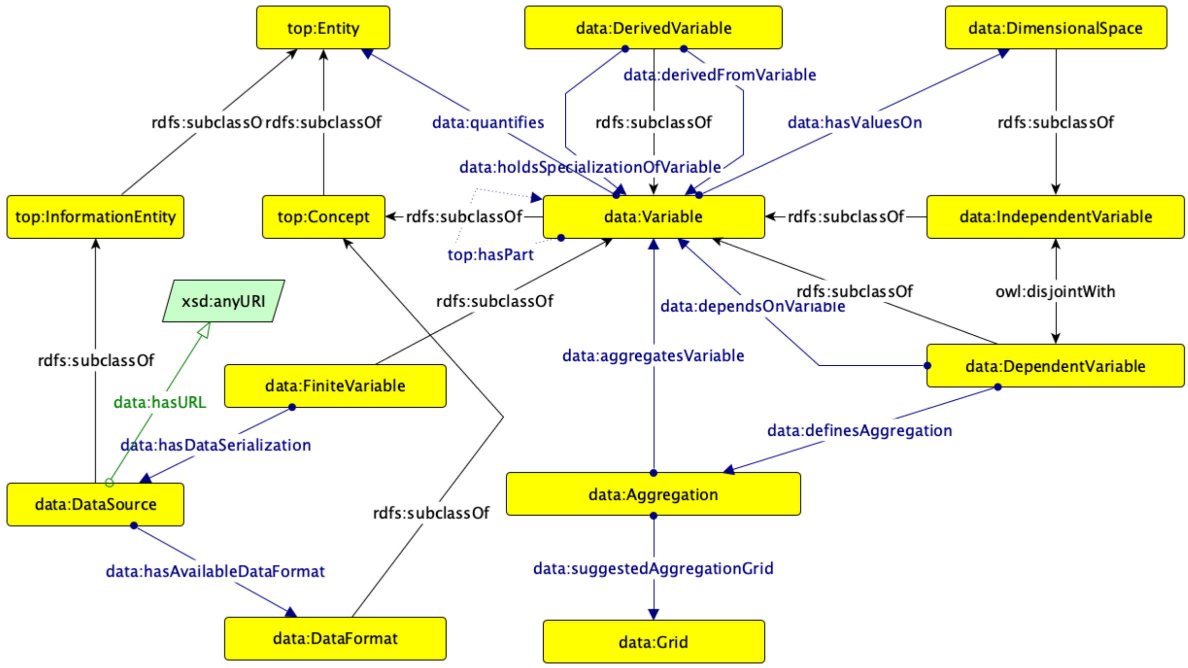


Figure 7. Representation of Variables and their relations with Data Sources.

A data:DimensionalSpace is composed of a set (not necessarily finite) of regions (data:Region). It can be either continuous (data:Continuum) or discrete (data:DiscreteDimensionalSpace). A discrete dimensional space can be a grid (data:Grid), i.e. composed of a set of regions that form a partition of all the space.

In order to avoid unnecessary complexity in concrete usage, a data:DimensionalSpace is also a data:IndependentVariable. Conceptually, it associates a dimensional space with a "default variable" with values on that dimensional space (on itself). This allows using directly a dimensional space as a variable when there are no possible ambiguities: e.g., a variable representing time in a context where no other variables represent time, can be directly represented by the corresponding temporal dimensional space.

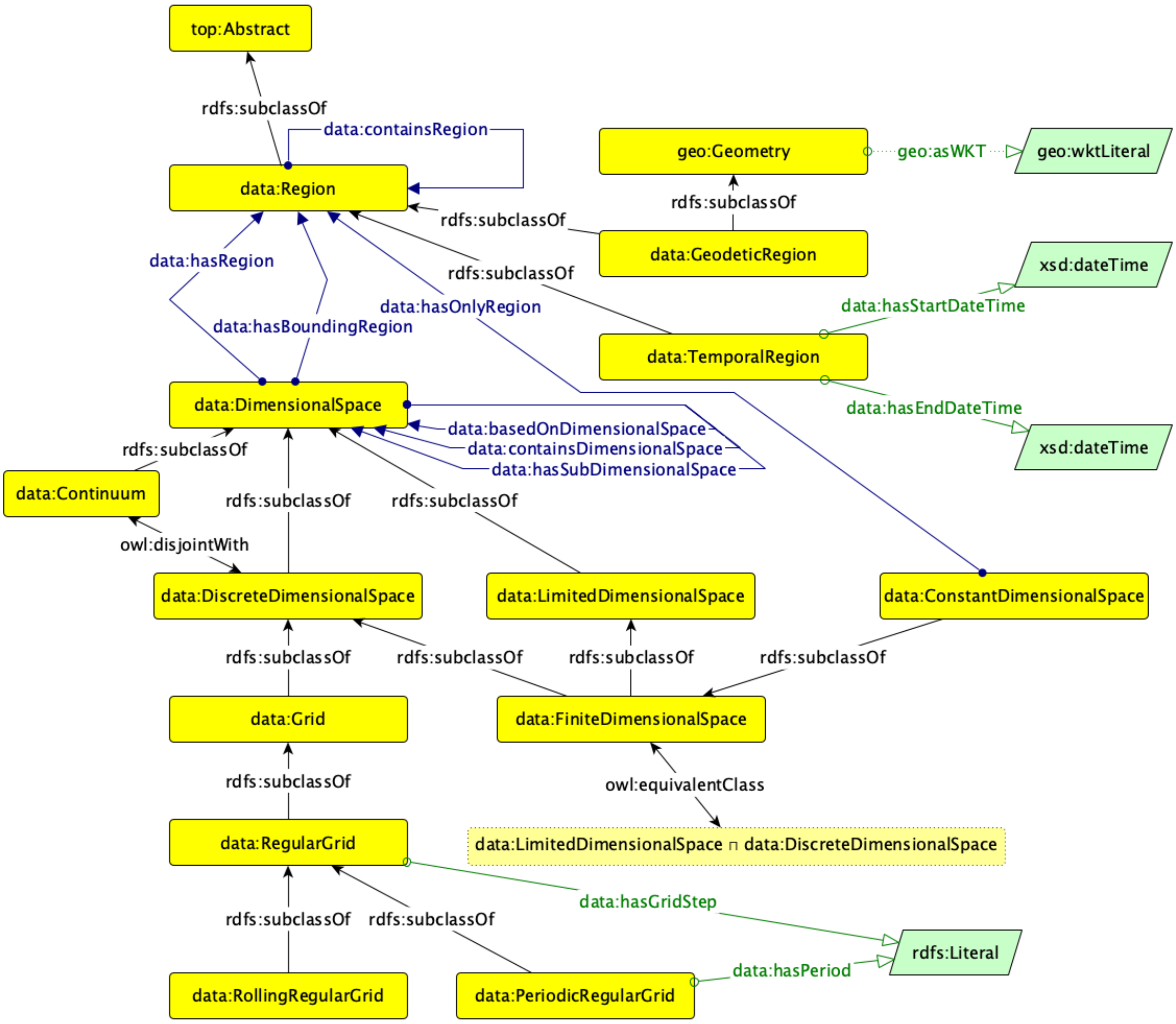


Figure 8. Representation of Dimensional Spaces.

The example graph shown in the diagram below contains the representation of a dataset (instance of `data:FiniteVariable`), its availability (via instances of `data:DataSource` and `data:DataFormat`), and the general variables (`mip-variable:tas`[10] and `mip-variable:tasmin`) which it specialises in the specific case of concern (e.g., the output of a simulation using a climate model). Furthermore, it further represents the "genealogy" of the variables: both `mip-variable:tas` and `mip-variable:tasmin` are aggregating variables derived from the variable tas representing the air temperature near the surface, which in turn is a specialisation of the general air temperature variable (`cf-standard-name:air_temperature`).

Each variable is associated with the corresponding dimensional spaces of the variable itself and the ones of the variables it depends on (which are in this case and many more time and space, represented with different dimensional spaces corresponding each one to specific reference frame, coverage, and granularity).

[10] While this not crucial to understand the example, the prefixes `mip-variable:` and `cf-standard-name:` are associated respectively to the namespaces `https://w3id.org/hacid/data/cs/variable/mip/` and `https://w3id.org/hacid/data/cs/variable/cf/`

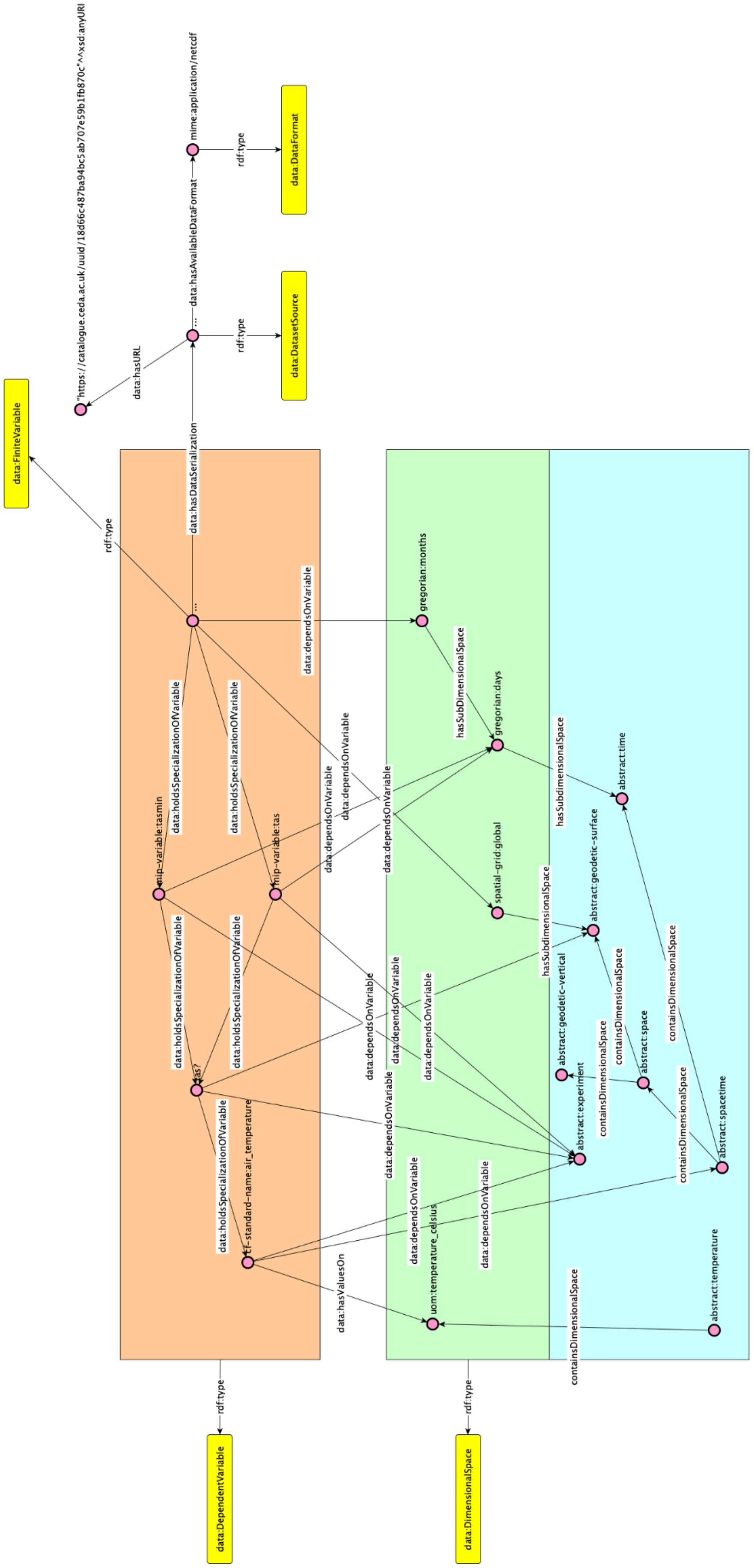


Figure 9. Representation of a climate dataset, with all the variables available or related.

Macroscopic Aspects: Climate Phenomena and Risk Assessment

Climate phenomena are a crucial part of the climate science discourse as they offer macroscopic abstractions of the small-scale physical atmospheric (but not only) phenomena at a level that is relevant for their impact on humans and human-related assets.

Instances `ccso:ClimatePhenomenon` can be used to refer to specific phenomena (e.g., the heatwave in Europe from 2003-07-29 to 2003-08-13, the “El Niño” Southern Oscillation, hurricane Beryl, etc.), while instances of `ccso:ClimatePhenomenonType` represent types of phenomena (e.g., heatwave, global oscillation, hurricane).

A climate phenomenon is a (climate-related) hazard if it impacts on humans and human assets in general (non-strictly-human assets, like biodiversity, can be considered as well). A `ccso:HazardType` is hence a `ccso:ClimatePhenomenonType` that can impact (property `ccso:hasPotentialImpactOn` AssetType) some kind of assets (`ccso:AssetType`).

In relationship to assets, risk assessment methods include the evaluation of hazards, exposure, and vulnerability. The way hazards are represented has already been described. The exposure is represented by the impacts (`ccso:Impact`) that a hazard has (or is predicted to have according to some prediction) on some assets. Finally, a vulnerability (`ccso:Vulnerability`) is a specific quality of an asset that renders it vulnerable to one or more hazards.

For all the concepts described (climate phenomenon, impact, asset, vulnerability) the ontology provides a class for specific instances and a class for types. This choice allows the description of these risk factors in a specific case as well as in general terms. Controlled vocabularies are expected to be used for the type classes.

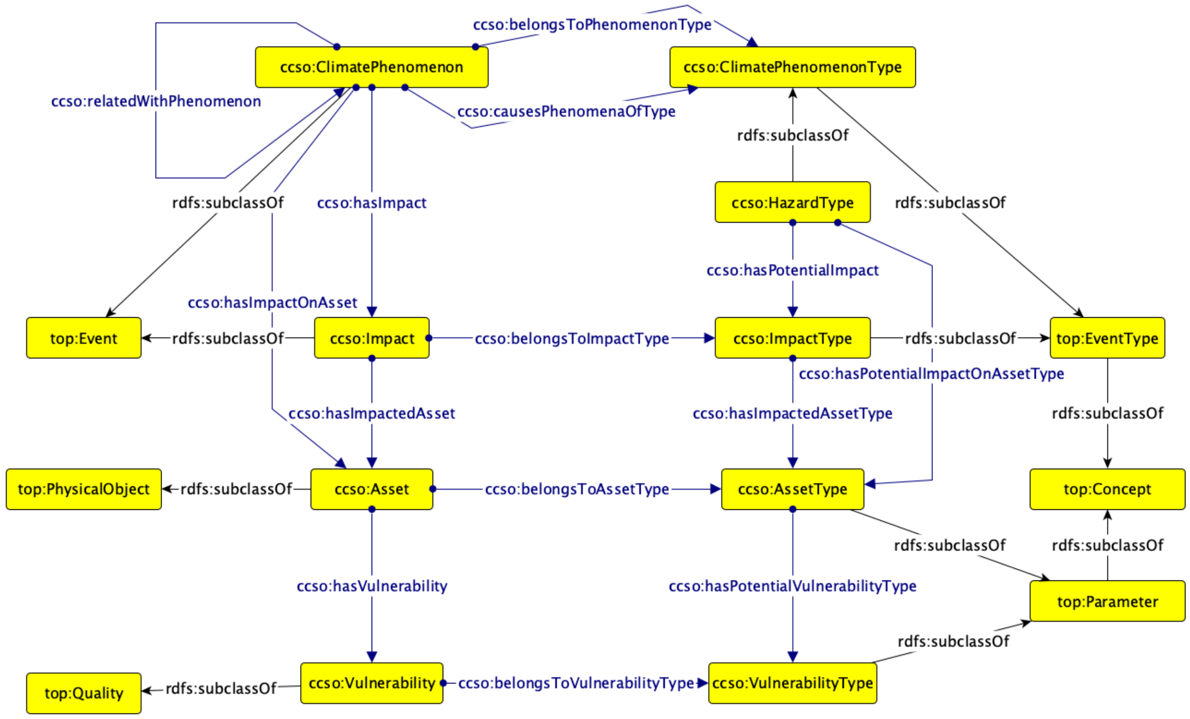


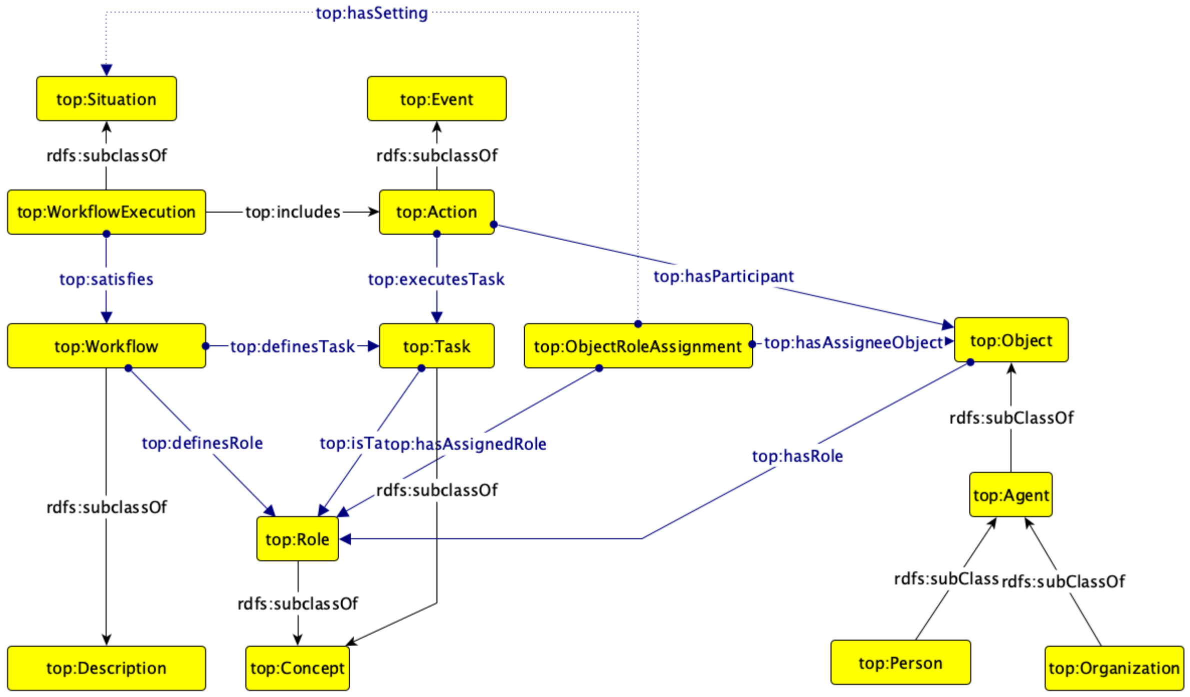
Figure 10. Representation of climate phenomena and risk assessment.

Climate Service Handling as a Process: Climate Workflows, Tasks, Roles, and Actions

Based on existing requirements and further interaction with domain experts, we identified the need to formally define aspects of the process of addressing a climate service case, which in some ways can be considered a process of data analysis, albeit influenced by specific context and experts' decisions that are connected to existing domain knowledge (both formal and informal). This knowledge is represented in the form of a workflow, where there are specified the tasks and the order in which they should be executed, and the role of the agent that executes the task. Such a workflow is expected to capture the process to arrive at a solution of a given climate service case, therefore modelling both the formalisation of the climate case requests and the contribution from experts to address the requests.

As introduced in Section 2.2.2, the representation of workflows is based on a pattern gathered from DOLCE[11] UltraLight [4], where workflows are considered descriptions and workflow executions are situations that satisfy those descriptions. Tasks and roles are concepts defined in the context of a workflow. Each time a workflow is executed, actions are performed to execute the generic tasks. These actions are participated by objects that fulfill roles established in the workflows. A specific subset of the defined roles are the ones to which tasks are assigned. For each workflow execution, those roles are associated with the actual agents (either people or organisations) executing the actions.

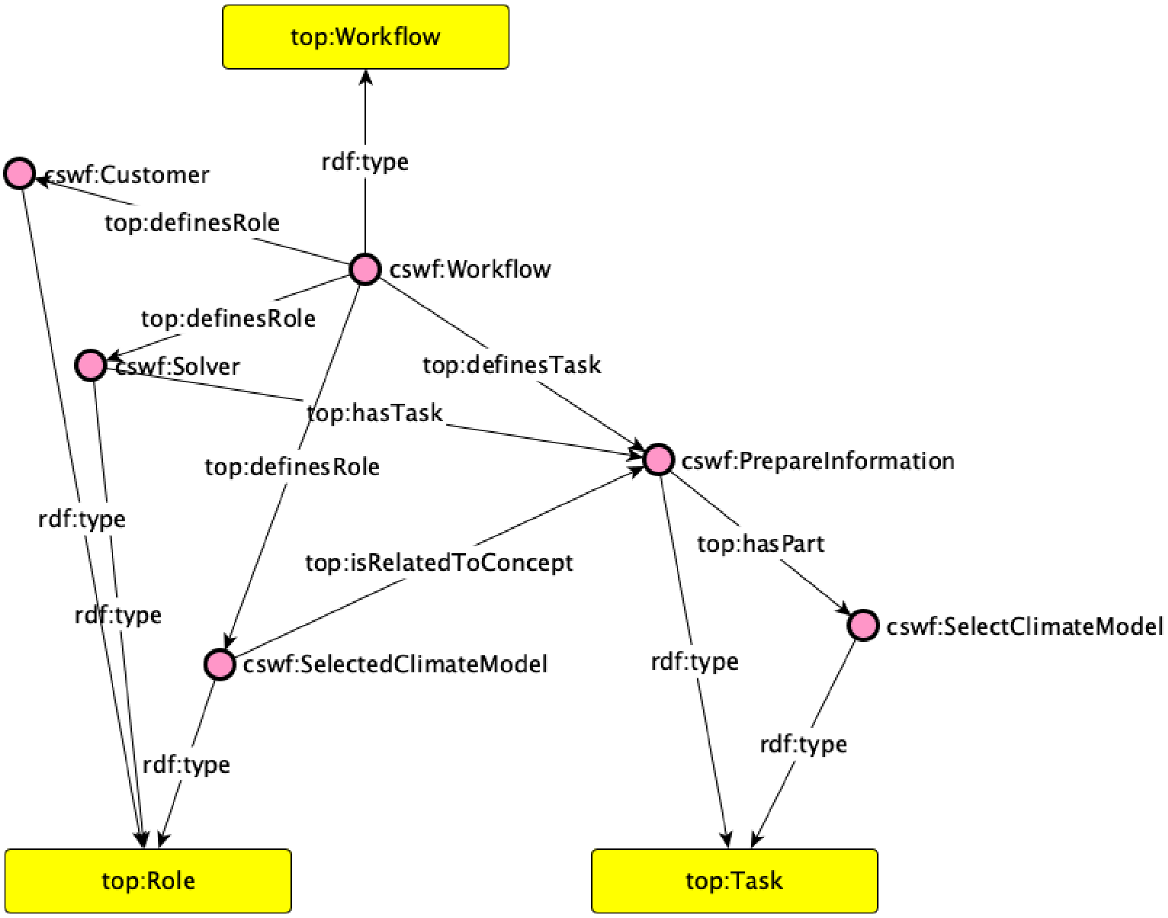
The association between roles and objects is also useful to link to the workflow resources relevant to some parts of the process (e.g., the hazards identified in the "identify hazards and asset sensitivity" task). Furthermore, we want to represent what conceptually amounts to a ternary relationship, i.e. the association between a workflow execution, a role, and an entity. For that reason we defined the class `top:ObjectRoleAssignment` that is a reification of such ternary relationship (see Figure 11).



[11] http://www.ontologydesignpatterns.org/ont/dul/DUL.owl

Figure 11. Ontology terms describing workflows and their executions.

Figure 12 shows an example of how a workflow can be specified. While the representation is obviously partial for conciseness reasons, all the presented items come from a detailed structural description of the typical workflow adopted for addressing a climate case (elicited with domain experts). The node `cswf:Workflow` [12] identifies such workflow as instance of `top:Workflow`. The workflow defines a number of roles, including roles that need to be fulfilled by agents (e.g., `cswf:Customer`, `cswf:Solver`) as well as roles that are fulfilled by other types of objects (e.g., `cswf:SelectedClimateModel`). The workflow also defines tasks (e.g., `cswf:PrepareInformation`), which are further hierarchically organised in subtasks (e.g., `cswf:SelectClimateModel`). Tasks are associated to roles that are relevant to their realisation: specifically, in this example, the task `cswf:PrepareInformation` is assigned to the role `cswf:Solver` (using the specific property `top:hasTask`) and the task `cswf:SelectClimateModel` is related to the role `cswf:SelectedClimateModel` (using the generic property `top:isRelatedToConcept`) because the former is meant to produce the latter[13].



[12] In this example and the following one, the prefixes `cswf:` and `model:` are associated with appropriate sub namespaces of the general one for instances in the climate service knowledge graph (https://w3id.org/hacid/data/cs/). The prefix `ex:` is used for instances of a specific case which are not part of the domain knowledge graph.

[13] Specialised properties may be defined to represent specific relationships like this one, but it is non trivial to do a relevant and coherent classification of such cases, so this is not currently formalised in our ontology.

Figure 12. RDF graph (partially) representing a workflow specification.

Figure 13 shows a partial representation of the execution of a workflow. The node `ex:wf_execution_42` identifies such execution as a whole, while nodes like `ex:model_selection_25` (related to the workflow execution by the `top:includes` property) identify actions, i.e., the execution of specific tasks of the workflow. The property `top:satisfies` associates the workflow execution to the workflow, while the property `top:executesTask` associates an action to the executed task. Instances of `top:ObjectRoleAssignment` (left unnamed in this example) are used to associate objects to workflow roles in the context of this execution: in this case `ex:JohnSmith` and `model:HadCM3` fulfill respectively the roles `cswf:Solver` and `cswf:SelectedClimateModel`. Furthermore, objects involved in an action are directly associated to it through the `top:hasParticipant` property (in this case `ex:JohnSmith` and `model:HadCM3` are both involved in the `ex:model_selection_25` action).

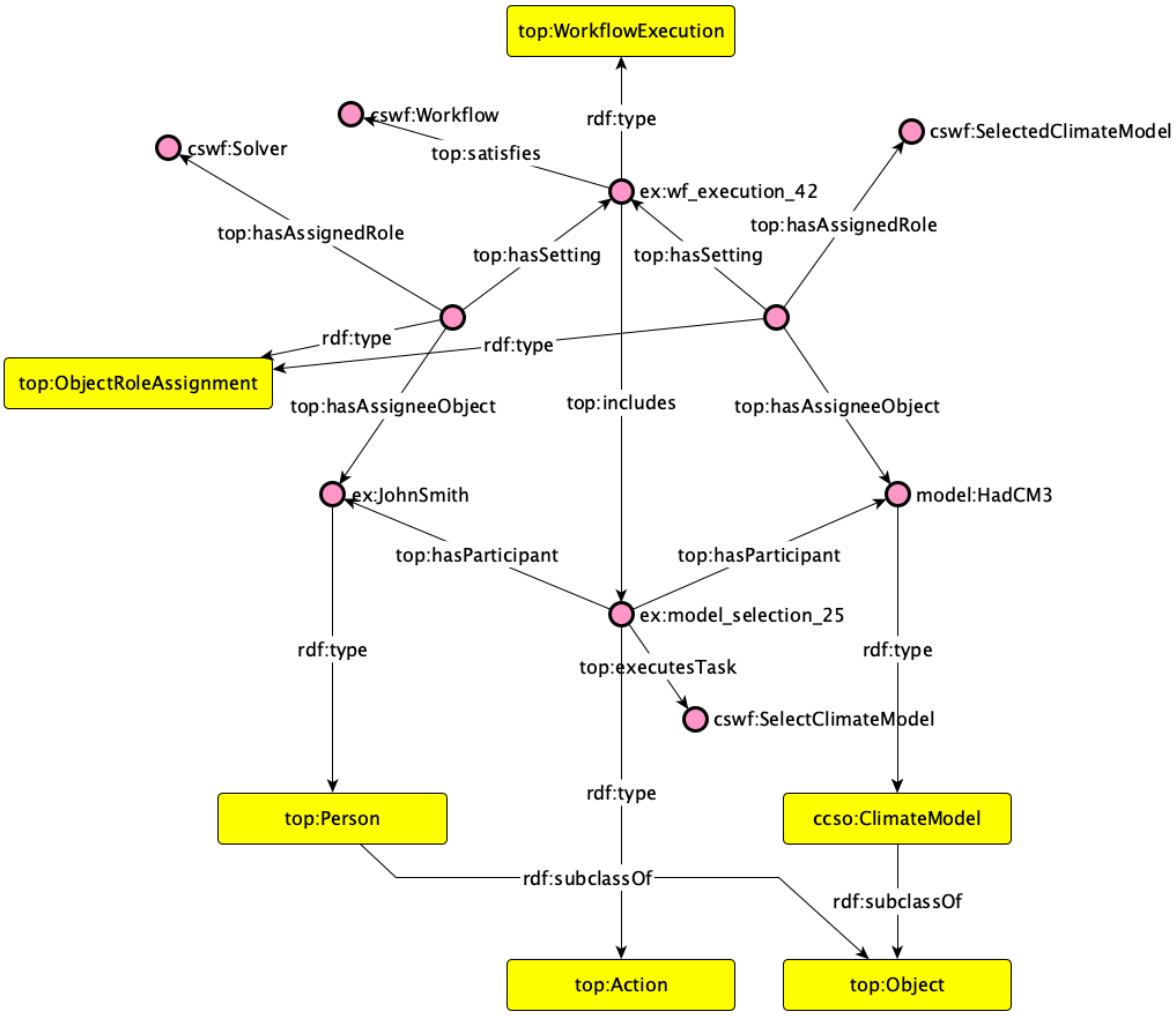


Figure 13. RDF graph (partially) representing a workflow execution.

2.2.4.2. Data

The data sources described in Section 2.1.2 have been used to produce RDF data modelled according to the ontology network presented in Section 2.2.4.1. In the next paragraphs we detail the methodology, its implementation, and obtained results.

2.2.4.3. Methodology

In order to map the resources to RDF (and precisely to the defined ontology), we adopted a three-step data handling process: first data cleansing, then mapping, and finally consolidation. Such a multi-step paradigm enables us to solve issues that are cumbersome or unfeasible to be dealt with by a mapping language alone.

Below a high level view of the adopted process is described with pseudocode.

```
# Non-automated part

for each datasource:
      gather the relevant structured or semi-structured data;
store them as local files;

# Semi-automated part

prepare an empty dataset on a local triple store;

for each datasource:
      ####### 1. DATA CLEANSING #######
      perform data cleansing (if needed) on the original files;
      reformat (if needed) to help mapping (e.g., from JSON to CSV);
      ####### 2. MAPPING #######
      map to RDF;
      upload RDF to local dataset;

####### 3. CONSOLIDATION #######
for each consolidation task:
      run on the local dataset a SPARQL update performing the task;

export local dataset as RDF;
publish novel/updated RDF dataset on public-facing triple store;
```

For every data source and every step of the process we automatised the procedure as much as possible and adopt existing standards, specifically declarative mapping/transformation languages. The main technology adopted for mapping to RDF is the RDF Mapping Language (RML)[14], which allows for multiple input formats (CSV, JSON, XML). RML itself is expressed in RDF, therefore potentially allowing inclusion in knowledge graphs for interoperability and full integration with the mapped data (e.g., to operationally represent provenance). Furthermore, RML offers an extension point in the possibility to include user

[14] https://rml.io/

defined functions, defined in some host language (in our case, in Python); while Python functions reduce the reusability of the RML mapping in other languages, they are necessary to achieve some of the required transformations.

For some data sources, data cleansing is needed before the actual mapping. Some cleansing is dealt by hand (if it is a very simple change), while for some sources in the JSON format ***jq***[15] is used. The ***jq*** language is a declarative language broadly adopted to perform JSON to JSON transformations. Furthermore, as RDF can be represented in JSON as JSON-LD, ***jq*** can be used also to perform the mapping from JSON to RDF (by generating JSON-LD). In some cases this method gives a more concise representation of the mapping and it is hence adopted for that purpose.

Regarding the consolidation tasks, those are additions to the knowledge graph that can be performed only after the mapping because they employ some form of aggregation of the mapped data. They are represented as SPARQL updates of the form `INSERT {...} WHERE {...}`. Currently there is only one consolidation task that we detail in the subsequent Section 2.2.4.4.

2.2.4.4. Implementation

All the code used to map the data to the climate services knowledge graph is publicly available as part of the GitHub repository with code related to the HACID ontologies and knowledge graphs[16]. The code (a part from the declarative bits in RML, ***jq***, and SPARQL Update) is mainly Python, using pyRML[17] for execution of RML mappings and other libraries for specific data conversions (e.g., managing dates and times). In addition there are some shell scripts that interact with the triple store or execute ***jq*** mappings. The triple store adopted for the mapping phase is Apache Jena Fuseki[18].

All the code related to the mappings is available in the `data/climate-services/` folder of said repository[19]. Under the folder there is a subfolder for each mapped datasource (e.g., `cmip5`, `cmor-tables`, ...). For each of those subfolders there are at least three subfolders: `data`, containing the data files in their original format; `rml` or `mapping`, containing the mapping(s) from the data to RDF; `rdf`, containing the RDF output of the mapping. If pre-processing (data cleansing) is needed there is a `data-raw` subfolder containing the actual initial files (in that case the files in `data` are the result of the cleansing process). For reasons of space the data files (both the original ones and the generated RDF files) are not uploaded to the the repositories, with the exception of small datasets. All the input data can be nevertheless obtained from public online sources and the output RDF generated by running the mappings.

In the `data/climate-services/` folder there are also the files to run the main steps of data handling process on all the data sources at once:

- `ws-data-rml.py` is the Python script that executes the RML mappings for all the subfolders;

[15] https://jqlang.github.io/jq/
[16] https://github.com/hacid-project/knowledge-graph
[17] https://github.com/anuzzolese/pyrml
[18] https://jena.apache.org/documentation/fuseki2/
[19] https://github.com/hacid-project/knowledge-graph/tree/main/data/climate-services

- `graph_names.csv` is the list of all the produced RDF files, associated with the corresponding graph name in the output RDF dataset;
- `upload_kg.sh` is a shell script loading all the RDF files in the local triple store, using the information in graph_names.csv;
- `simTimeCoverage.ru` is a SPARQL Update performing the consolidation task of computing the time coverage of simulations based on the aggregated time coverage of the corresponding datasets.

The data cleansing step, previous to the other ones, is currently managed case by case when needed (either performing some data handling by hand or in one case using jq and shell script). Furthermore, also for the mapping performed for the climdex data, there is an independently run shell script calling ***jq***.

2.2.4.5. Example Datasource Mapping: CORDEX datasets

As an example of a mapped data source, let consider CORDEX datasets. CORDEX datasets are output of the regional dynamical downscaling of global climate simulations. All the specific code and data is under the `data/climate-services/cordex/` folder. In the `raw-data` directory there will be the original files. In this case they have to be retrieved via web API, specifically from the CEDA node of ESGF[20]. They have to be retrieved through multiple calls, as the number of results (162,251) is higher than single call limit results (10,000). In order to retrieve them, in `raw-data` there is a shell script (`download-data.sh`) that performs all the required calls and stores the results in separate JSON files under the `raw-data/chunks/` folder.

Then the data must be cleansed using a script in the top level of the CORDEX folder (`clean-data.sh`). The script performs a ***jq*** transformation defined in another file (`clean-data.jq`) that manipulates the fields related to the driving model: the driving model is the one used for the original global simulation, but in the original data it is mixed up with the corresponding institution and has to be fixed in order to maintain coherence with the other data. This transformation is applied to every file in the `raw-data/chunks/` folder and the results are stored (still as separate files, to avoid overloading memory) in the `data/` folder.

Then the main mapping is performed using RML. The mapping is executed by the previously introduced common Python script `ws-data-rml.py` that performs the RML mapping for all the datasources. Its behavior is configurable for each datasource through a `conf.json` file under the corresponding directory (directories that do not contain such file are ignored). For JSON input files (as it is often the case) the Python code by default converts it to CSV before applying the RML mapping, as this often simplifies the RML structure. The `conf.json` file specifies (in the content of the `jsonpath` key) a JSONPath expression to be used to get the sequence of JSON objects that will become the rows of the CSV; for each selected JSON object a row will be generated, having as values all the values of the corresponding JSON keys. For the CORDEX datasets case the JSONPath expression is `$.response.docs[*]`, which evaluates to the sequence of JSON objects corresponding

[20] https://esgf.ceda.ac.uk/esg-search/search/

to all the datasets in the API response. The CSV version of all the files is stored again in the `data/` folder, alongside the JSON files.

As a second step, `ws-data-rml.py` performs also the actual RML mapping. For that it looks for an RDF file under the `rml/` subfolder of each datasource: for the CORDEX case it is `rml/cordex.ttl`. The RML mapping in this case describes how each CORDEX dataset and its associated resources (e.g., the simulation) are associated to URIs, how they are related by triples between each other and to literals. Special care is given to the design of the URI schemes for related resources as the simulations, so that a certain simulation always corresponds to the same URI, even if it gets created again for each different corresponding dataset. If multiple different URIs would be generated for a single concept, the knowledge graph would be redundant and semantically incoherent. The RDF output of the mapping is stored as Turtle files in the `rdf/` folder.

As described in Section 2.2.4.3, the RDF files are then loaded in a local triple store for consolidation. For that purpose we adopted Apache Jena Fuseki, specifically using TDB2 persistent storage. A new dataset has to be created (using the included web-based graphical user interface it is easy to drop the dataset with the previous version and create a new one). Then the general `upload_kg.sh` script can be used to load all the RDF files to the new dataset[21], using the file paths and named graph URIs defined in the `graph_names.csv` file. RDF data about CORDEX datasets is placed in a specific named graph[22].

At this point consolidation can be performed in the local triple store in the form of SPARQL Update operations. Currently there is only one SPARQL Update file, with a single operation: `simTimeCoverage.ru`. Finally, after consolidation the whole RDF dataset can be exported in some RDF serialisation supporting named graphs (e.g., N-Quads), ready to be loaded on a public facing triple store in order to expose the updated version of the HACID climate services knowledge graph.

2.3. Testing and evaluation

Testing and evaluation are critical processes in the development and maintenance of the domain knowledge graphs, serving as essential mechanisms to ensure their reliability, accuracy, practical utility and relevance to the target domains of application. Consolidated assessment methodologies help validate the structural integrity, semantic coherence, and information quality of KGs by systematically examining various dimensions such as completeness, consistency, correctness, and coverage. Without comprehensive testing and evaluation, KGs risk containing inaccurate entities and/or relationships, incomplete information, or semantic inconsistencies that could compromise their effectiveness in supporting data integration as well as reasoning and knowledge inference. Consequently, robust evaluation frameworks not only help identifying potential errors and gaps, but also provide insights into the KGs' performance, enabling assessing its representational and computational capabilities, to ensure they meet the desired requirements for real-world applications.

Following state of the art approaches and methodologies to KG validation [5], [6], [7] we evaluated our domain KGs along three core dimensions:

[21] For performance reasons, the script operates directly on the TDB2 storage files and hence the Fuseki server must be temporarily down so that the script can have exclusive access to the storage.
[22] https://w3id.org/hacid/ccso/data/cordex/datasets

1. **functional**, to assess the ability of a KG to address requirements and capture the knowledge it is supposed to model;
2. **logical**, to check whether the ontology networks can be successfully processed by a reasoner and inference over a KG produces the intended results;
3. **structural**, to assess the topological properties of a KG using context-free metrics, which analyze its graph-based representation independently of domain-specific semantics or external contextual information, focusing on topological features.

The *functional* and *logical* dimensions are crucial for evaluating the quality of a KG, in particular in terms of *compliance to expertise*, i.e., the property of the reference KGs to be compliant with the domain knowledge they are supposed to model. However, analyzing the *structural* dimension offers valuable insights into design choices, using indicators that can reveal potential strengths or weaknesses in quality. In addition, as discussed in Section 2.3.1 below, the *functional* evaluation directly contributes to the quantitative evaluation for the Key Performance Indicator "KPI-3 - Knowledge engineering", and in particular for "KPI-3b: KG coverage". In the rest of this section, we detail the evaluation performed across the three core dimensions and report on the relevant results.

2.3.1. Functional evaluation

The functional dimension pertains to the intended use of a KG and its components, focusing on their role within a specific application context. This dimension is fundamental to KG testing, as it enables designers to evaluate how effectively a KG meets defined requirements and comprehensively represents the target domain.

We adopt the testing methodology outlined in [5], [6], which emphasizes evaluating the alignment of a KG with its requirements, understood as the ontological commitments and knowledge representation requirements expressed through competency questions (CQs), i.e., the functional dimension. During the design phase, user stories are translated into one or more CQs, which in turn are associated with corresponding unit tests, which, when executed, help validate the KG. Functional evaluation of the KGs is thus concretely performed in the form of **CQ verification**. This approach involves testing whether the ontology networks enables the translation of a CQ, representing an ontology requirement, into a corresponding SPARQL query, which is then executed over the KG to verify whether it returns the expected answer, ensuring the KG effectively supports the requirement. The CQ verification process includes three fundamental steps:

1. **Defining and listing the Competency Questions (CQs)**
   CQs are formulated to capture the functional and knowledge representation requirements that guide the ontology design process; they act as benchmarks for what the ontology and KG are expected to support and answer.
2. **Formulating SPARQL queries and expected answers**
   For each CQ, a corresponding SPARQL query is created, designed to extract the relevant information from the KG using the ontology's classes and properties. Alongside, an expected result or answer is defined, representing the correct or desired response to the question.

3. **Executing queries and comparing results**
   Each SPARQL query is executed on the target KG (or on user-defined test RDF data), and the actual results are compared against the predefined expected answers. This comparison verifies whether the KG can satisfy the information needs expressed in the query.

This process serves two important purposes:

- It confirms that the ontology provides the necessary classes and properties to express the CQs as SPARQL queries. If a query cannot be formulated, it suggests that the ontology may be incomplete or misaligned with the requirements.
- By evaluating whether the KG provides answers that match the expected results, the process ensures that the data within the KG adequately supports the intended use cases and information needs. Discrepancies might indicate issues with the data's coverage, quality, or correctness.

As the XD (eXtreme Design) methodology [2] was followed in the ontology development process, CQs were used as foundational inputs during the design phase of all the ontology modules, as documented in Deliverable 2.1, ranging from the top-level and common modules to the domain specific modules for medical diagnostics and climate services. Specifically in D2.1 [1], for each ontology module, the fundamental CQs were explicitly identified and listed, and for many of them a corresponding SPARQL query was provided. While CQs and queries identified for the medical diagnostic domain were already consolidated in D2.1, CQs and queries for the climate services domain were subjected to a process of revision and extension, as part of the iterative and incremental design process. Additionally, minor revisions and extensions were applied to the top-level and common modules, as reflected in this document. To conduct a systematic evaluation of the KGs, all CQs were converted into SPARQL queries together with the expected result, and all of them were executed on the target KG, following the process outlined above. In line with the *agile* approach that characterises the XD methodology, this evaluation is performed iteratively throughout the development workflow, ensuring continuous validation. Whenever a query result does not match the expected outcome, it triggers a revision activity, prompting adjustments to the KG, queries, or underlying ontology as needed.

CQ verification test cases are defined using the OWLUnit Ontology[23]. By relying on this vocabulary, a test case can be specified in RDF defining the core elements that enable the specification and execution of the test case, i.e., the competency question in natural language (`owlunit:hasCompetencyQuestion`), the corresponding SPARQL query (`owlunit:hasSPARQLUnitTest`), the expected result (`owlunit:hasExpectedResult`) and the data or SPARQL endpoint (`owlunit:hasInputData`) to be used to evaluate the query and get the result to be compared with the expected one. An example is provided below, where a test case is first defined in a simple tabular format and then specified as an OWLUnit test case.

ID	mdx-1
CQ	What are the specialties of a given clinical case?

[23] https://w3id.org/OWLunit/ontology/

SPARQL query	PREFIX : <https://w3id.org/hacid/mdx/data/clinicalcase/>
	PREFIX mdx: <https://w3id.org/hacid/onto/mdx/>
	
	SELECT ?specialty
	WHERE {
	  :1883 a mdx:ClinicalCase ;
	        mdx:hasSpecialty ?specialty .
	  ?specialty a mdx:Specialty
	}

Endpoint	https://w3id.org/hacid/sparql
Expected result	● https://w3id.org/hacid/mdx/data/specialty/endocrinology ● https://w3id.org/hacid/mdx/data/specialty/neonatology-and-perinatology

```
@prefix owlunit: <https://w3id.org/OWLunit/ontology/> .
@prefix test: <https://w3id.org/hacid/data/testcase/> .

test:mdx-1.ttl a owlunit:CompetencyQuestionVerification ;
  owlunit:hasCompetencyQuestion "What are the specialties of a given
                                 clinical case?" ;
  owlunit:hasSPARQLUnitTest
   """
   PREFIX : <https://w3id.org/hacid/mdx/data/clinicalcase/>
   PREFIX mdx: <https://w3id.org/hacid/onto/mdx/>

   SELECT ?specialty
   WHERE {
     :1883 a mdx:ClinicalCase ;
           mdx:hasSpecialty ?specialty .
     ?specialty a mdx:Specialty
   }
   """ ;
  owlunit:hasInputTestDataCategory owlunit:SPARQLEndpoint ;
  owlunit:hasInputData <https://w3id.org/hacid/sparql> ;
  owlunit:hasExpectedResult """
    {
      \"head\": {
        \"vars\": [ \"specialty\" ]
      } ,
      \"results\": {
        \"bindings\": [
         {
          \"specialty\": {
            \"type\": \"uri\" ,
            \"value\":
\"https://w3id.org/hacid/mdx/data/specialty/endocrinology\"
          }
        },
        {
          \"specialty\": {
            \"type\": \"uri\" ,
            \"value\":
\"https://w3id.org/hacid/mdx/data/specialty/neonatology-and-perinatology\"
         }
       }]
      }
    }""" .
```

Test cases defined using the OWLUnit Ontology are the input to the OWLUnit tool[24] which enables parsing these test cases, executing them, and evaluating the results programmatically. Defining ontology test cases as RDF data using OWLUnit ensures a robust, interoperable, and semantically rich approach to ontology validation, facilitating automated testing processes and enabling continuous integration workflows where ontology updates are automatically tested to ensure they meet defined requirements.

While providing a comprehensive list of all test cases is beyond the scope of this document, the complete test suites are available in dedicated sections of the project's GitHub repository on knowledge graphs[25]. These test cases were defined for the different ontology modules based on the set of competency questions identified in D2.1 and further consolidated during the ontology development process. In line with the modular design approach adopted for the ontology, each module has its own dedicated test suite containing test cases. This modular organization ensures that each ontology module can be independently validated against its specific requirements. Each module-specific folder with test cases includes detailed descriptions, RDF representations, test data and expected outcomes for each test case, as well as instructions on how to run test cases with the OWLUnit tool.

More in detail, in the reference GitHub repository, each directory (under the `ontologies` directory) that contains the OWL specification of one or more ontology modules also includes a `test` directory where test cases for functional and logical evaluation of that module are defined. So for example the `ontologies/top-level/test` directory includes test cases for the top-level ontology module.

As CQ verification test cases directly relate to competency questions, the number of CQ verification test cases defined per ontology module reflects the number of CQs defined for each module. The number of test cases per ontology module is summarised in Table 9, on the basis of CQs defined in D2.1 and, where needed, in this document (cf. Section 2.2.2 where additional CQs for the top-level modules are provided).

Table 9. CQ validated for each ontology module.

Ontology module	CQ verification test cases	Passed test cases
`top`	23	23
`agentrole`	3	3
`judgement`	5	5
`evidence`	3	3
`naming`	3	3
`mdx`	7	7
`ccso`	8	8

[24] https://github.com/luigi-asprino/owl-unit
[25] https://github.com/hacid-project/knowledge-graph

Ontology module	CQ verification test cases	Passed test cases
Total	52	52

As already mentioned above in the introduction to Section 2.3, the functional evaluation process, specifically in the form of CQ verification tests, directly contributes to the quantitative evaluation for “KPI-3b: KG coverage” in the context of “KPI 3 - Knowledge engineering”. KPI-3b, defined as “Accuracy@10 ≥ 0.75 for competency questions (CQs) converted to SPARQL queries”, basically requires that for at least 75% of the SPARQL queries derived from CQs the expected answer is within the first 10 results. In essence, 75% of the CQ verification test cases must be passed and successfully executed, with the SPARQL query producing the expected answer.

KPI-3b permits partial success, allowing for some unanswered queries. However, the eXtreme Design ontology design methodology we follow prioritises completeness and correctness. This means that every CQ within the knowledge graph’s scope must not only be representable as a SPARQL query but also yield an answer that matches the expected one. In essence, all functional requirements of the ontology (as captured by CQs) must be fully satisfied, meaning that all CQ verification test cases must be passed. Test results reported before, with all test cases successfully passed, yields to Accuracy@10 = 1 for KPI-3b.

2.3.2. Logical evaluation

The logical dimension of the testing and validation approach evaluates whether an ontology can be successfully processed by a reasoner, to ensure that the ontology adheres to logical consistency and reasoning capabilities. This dimension is crucial especially when ontologies are used to enable automated reasoning, where a reasoner (such as a classifier or inference engine) processes the ontology to infer new knowledge, validate relationships, or identify inconsistencies in a KG. Overall, assessing whether the ontology network can be processed successfully and efficiently by a reasoner provides an indication about its computational integrity and overall efficiency.

To ensure the semantic coherence and logical consistency of the ontologies in the networks, the evaluation process begins by using a reasoner to check the ontologies for any inconsistencies. The target of this step is to confirm that no inconsistencies are detected, thereby establishing that the ontology axioms are logically coherent and consistent. Ontology editors, and in particular the Protégé ontology editor used in the project, provide built-in support for using a reasoner to evaluate ontologies for inconsistencies. This functionality enables users to check whether the logical structure of an ontology is consistent and coherent. When developing ontology modules in Protégé, a reasoner was periodically activated by the designer to analyze the ontology’s axioms, detect contradictions, and identify issues like unsatisfiable classes or conflicting constraints.

Logical evaluation then includes two additional testing and validation scenarios: *(i)* inference verification and *(ii)* error provocation.

Inference verification. Inference verification is conducted by applying a reasoner to a given data sample in conjunction with the ontologies. This step checks whether the inferred knowledge aligns with the expected inferences, ensuring that the ontology axioms accurately reflect the intended knowledge representation requirements and constraints. Discrepancies between the actual inferences and the expected results highlight potentially missing axioms and constraints in the ontology.

Error provocation. Error provocation is employed to test the reasoner's ability to detect inconsistent data that violates the intended constraints defined in the ontology. Synthetic data is generated with intentional logical inconsistencies relative to the ontology axioms. These inconsistencies are then used to determine whether the reasoner can successfully identify and flag them. If an expected logical inconsistency is not detected by the reasoner, this suggests that the appropriate axioms (e.g., disjointness axioms) to capture the domain requirement and constraint are missing.

For both inference verification and error provocation, structured test cases can be defined and executed. To this end, we rely again on the OWLUnit ontology, which allows defining `owlunit:InferenceVerification` and `owlunit:ErrorProvocation` test cases as RDF specifications to be parsed, executed and evaluated by the OWLUnit tool, as already explained in the previous section for CQ verification test cases.

More in detail, an inference verification test case is specified by defining:

- the ontology to be tested;
- the data sample to be used for the evaluation;
- the reasoner to be employed;
- a SPARQL query to be executed on the KG generated after applying the reasoner to the data sample according to the ontology axioms; the query is designed to check for the presence of the expected inferred knowledge resulting from the reasoning step;
- the expected result, which will be compared with the actual output produced by the SPARQL query.

When processing such a test case specification, the OWLUnit tool checks that: 1) the tested ontology is logically consistent; 2) the ontology and the input data sample together don’t lead to any inconsistency when applying the reasoner; and 3) the result of the SPARQL query is equivalent to the expected result.

As a simple example in the medical diagnostics domain, we consider the following ontology fragment from the ontology module derived from the SNOMED CT resource, stating that the class EvaluationProcedure is a subclass of the class Procedure (i.e., that all EvaluationProcedures are Procedures).

```
@prefix hsct: <https://w3id.org/hacid/onto/hsct/> .
@prefix owl: <http://www.w3.org/2002/07/owl#> .
@prefix rdfs: <http://www.w3.org/2000/01/rdf-schema#> .

hsct:EvaluationProcedure a owl:Class ;
                         rdfs:subClassOf hsct:Procedure .
```

Given a data sample as reported hereafter where an instance of the EvaluationProcedure class is defined (namely, “Magnetic resonance imaging of face and orbit with contrast”), we expect a reasoner to derive that such an individual is also an instance of the Procedure class.

```
@prefix hsct: <https://w3id.org/hacid/onto/hsct/> .
@prefix mdxdata: <https://w3id.org/hacid/data/mdx/> .
@prefix rdfs: <http://www.w3.org/2000/01/rdf-schema#> .

mdxdata:16324921000119103 a hsct:EvaluationProcedure ;
   rdfs:label "Magnetic resonance imaging of face and orbit with contrast
(procedure)@en-gb .
```

To check whether the inference process produces the expected knowledge, the following SPARQL query can be used:

```
PREFIX hsct: <https://w3id.org/hacid/onto/hsct/>
PREFIX mdxdata: <https://w3id.org/hacid/data/mdx/>

ASK { mdxdata:16324921000119103 a hsct:Procedure }
```

The result of this query can then be compared with the expected one (`true`) to check whether the test case is successfully passed. This example can be represented using the OWLUnit ontology as a `owlunit:InferenceVerification` as follows.

```
@prefix owlunit: <https://w3id.org/OWLunit/ontology/> .
@prefix test: <https://w3id.org/hacid/data/testcase/> .

test:iv-mdx1.ttl a owlunit:InferenceVerification ;
  owlunit:testsOntology <https://w3id.org/hacid/onto/hsct> ;
  owlunit:hasInputTestDataCategory owlunit:ToyDataset ;
  owlunit:hasInputData test:iv-mdx1-data.ttl ;
  owlunit:hasReasoner owlunit:HermiT ;
  owlunit:hasSPARQLUnitTest
   """
   PREFIX hsct: <https://w3id.org/hacid/onto/hsct/>
   PREFIX mdxdata: <https://w3id.org/hacid/data/mdx/>

   ASK { mdxdata:16324921000119103 a hsct:Procedure }
   """ ;
  owlunit:hasExpectedResult true .
```

Such a test case specification can be processed and executed by the OWLUnit tool as described before, using the HermiT reasoner[26] as stated in the test case definition.

As a more articulated example, we consider the following ontology fragment from the ontology module for representing clinical cases and medical diagnostics scenarios.

```
@prefix owl: <http://www.w3.org/2002/07/owl#> .
@prefix rdfs: <http://www.w3.org/2000/01/rdf-schema#> .
@prefix top: <https://w3id.org/hacid/onto/top-level/> .
@prefix mdx: <https://w3id.org/hacid/onto/mdx/> .
```

[26] http://www.hermit-reasoner.com/

```
@prefix jdg: <https://w3id.org/hacid/onto/common/judgement/> .

mdx:isInScopeOfDiagnosis a owl:ObjectProperty ;
  rdfs:subPropertyOf top:associatedWith ;
  rdfs:domain mdx:DisorderDescription ;
  rdfs:range mdx:Diagnosis ;
  rdfs:comment "The relation for linking a disorder description to a
                diagnosis."@en .

mdx:hasDiagnosis a owl:ObjectProperty ;
  rdfs:subPropertyOf jdg:hasJudgmentResult ;
  rdfs:domain mdx:ClinicalEvaluation ;
  rdfs:range mdx:Diagnosis ;
  rdfs:comment "The property to associate a clinical evaluation the its
                resulting diagnosis."@en .

mdx:forClinicalCase a owl:ObjectProperty ;
  rdfs:subPropertyOf jdg:isJudgementOn ;
  rdfs:domain mdx:ClinicalEvaluation ;
  rdfs:range mdx:ClinicalCase ;
  rdfs:comment "The relation for associating a clinical evaluation with a
                clinical case."@en .

top:isSatisfiedBy a owl:ObjectProperty ;
  owl:propertyChainAxiom (
   mdx:isInScopeOfDiagnosis
   [ owl:inverseOf mdx:hasDiagnosis ]
   mdx:forClinicalCase
 ) .
```

The above fragment includes axioms to define a property chain in relation to the `top:isSatisfiedBy` object property, defined in the top level ontology module as the property for asserting that a description is satisfied by an eventuality or situation (in the sense of the DnS - Descriptions and Situations knowledge representation pattern). Basically, the property chain axiom aims at expressing that: (i) if a DisorderDescription *desc* is in the scope of a Diagnosis *diag* and (ii) a ClinicalEvaluation *eval* has *diag* as diagnosis and (iii) the ClinicalEvaluation *eval* is associated with a ClinicalCase *case*, then we can infer that the DisorderDescription *desc* `top:isSatisfiedBy` the ClinicalCase *case*.

Given a data sample as reported hereafter, we expect a reasoner to derive that `:desc top:isSatisfiedBy :case`. Beyond the knowledge that can be inferred from the property chain axiom, additional facts are expected to be produced by the reasoner, e.g., exploiting the axioms defining domain and range for the properties. Therefore, `:desc` is expected to be classified as a `mdx:DisorderDescription`, `:diag` as a `mdx:Diagnosis`, `:eval` as a `mdx:ClinicalEvaluation` and `:case` as a `mdx:ClinicalCase`.

```
@prefix mdx: <https://w3id.org/hacid/onto/mdx/> .
@prefix : <https://w3id.org/hacid/data/example/> .

:desc :isInScopeOfDiagnosis :diag .
```

```
:eval :hasDiagnosis :diag .
:eval :forClinicalCase :case .
```

This test case scenario can be again represented using the OWLUnit ontology as a `owlunit:InferenceVerification` as follows.

```
@prefix owlunit: <https://w3id.org/OWLunit/ontology/> .
@prefix test: <https://w3id.org/hacid/data/testcase/> .

test:iv-mdx.ttl a owlunit:InferenceVerification ;
  owlunit:testsOntology <https://w3id.org/hacid/onto/mdx> ;
  owlunit:hasInputTestDataCategory owlunit:ToyDataset ;
  owlunit:hasInputData test:iv-mdx-data.ttl ;
  owlunit:hasReasoner owlunit:HermiT ;
  owlunit:hasSPARQLUnitTest
   """
   PREFIX mdx: <https://w3id.org/hacid/onto/mdx/>
   PREFIX top: <https://w3id.org/hacid/onto/top-level/>
   PREFIX : <https://w3id.org/hacid/data/example/>

   ASK { :desc top:isSatisfiedBy :case }
   """ ;
  owlunit:hasExpectedResult true .
```

As already explained, this test case specification can be processed and executed by the OWLUnit tool, which uses the HermiT reasoner and executes the provided SPARQL query to check whether the expected additional knowledge (`:desc top:isSatisfiedBy :case`) is present in the KG after the reasoning step.

As an example of an error provocation test case, we consider the climate services domain. As explained in Section 2.2.4.1, variables play a fundamental role and can be classified along various dimensions and perspectives, including the distinction between *dependent* (`data:DependentVariable`) and *independent* (`data:IndependentVariable`) variables. It is reasonable to expect that these two concepts do not overlap, i.e., they are disjoint, so that a variable can not be both a dependent and an independent variable.

To validate the ontology regarding this requirement and thus check whether such a constraint is captured in the reference ontology module, we define a data sample as reported hereafter, where an entity is declared as being both a `data:DependentVariable` and a `data:IndependentVariable`.

```
@prefix data: <https://w3id.org/hacid/onto/common/data/> .
@prefix : <https://w3id.org/hacid/data/example/> .

:var a data:DependentVariable, data:IndependentVariable .
```

Running a reasoner over this data should result in an inconsistency being detected, under the assumption that the reference ontology module includes a disjointness axiom for the `data:DependentVariable, data:IndependentVariable` classes.

This test case scenario can be represented using the OWLUnit ontology as a `owlunit:ErrorProvocation` as follows.

```
@prefix owlunit: <https://w3id.org/OWLunit/ontology/> .
@prefix test: <https://w3id.org/hacid/data/testcase/> .

test:ep-cs-vars.ttl a owlunit:ErrorProvocation ;
  owlunit:testsOntology <https://w3id.org/hacid/onto/common/data> ;
  owlunit:hasInputTestDataCategory owlunit:ToyDataset ;
  owlunit:hasInputData test:ep-cs-vars-data.ttl ;
  owlunit:hasReasoner owlunit:HermiT .
```

For error provocation test cases, only the ontology to be tested and the inconsistent data sample are needed, as the goal is to detect the inconsistency in the data with respect to the ontology.

As for CQ verification tests, providing a comprehensive list of all test cases for inference verification and error provocation is beyond the scope of this document; the complete test suites are available in dedicated sections of the project's GitHub repository on knowledge graphs, together with test execution instructions. As already explained, the test-driven eXtreme Design methodology mandates that all test cases must be successfully passed in order to release an (version of) ontology module. All identified test cases focusing on inference verification and error propagation were successfully passed as part of the iterative design→test and fix→release XD process.

2.3.3. Structural evaluation

The evaluation of the functional and logical perspective presented in the previous sections aims at assessing the so-called *compliance to expertise* of the KGs, i.e., the property of the reference ontologies to be compliant with the knowledge they are supposed to model.

To evaluate the structural dimension of the KGs, we employ various metrics that have been defined and used in the literature [7], [8], [9], [10], [11]. This involves calculating base metrics that capture quantitative aspects of the KGs, including classes, their instances, properties, and axioms. Additionally, we compute schema and graph metrics to evaluate: *(i)* schema-level attributes such as richness, breadth, depth, and inheritance, and *(ii)* ontology-level characteristics including cohesion, coupling, multi-hierarchical degree, and extensional coverage.

As defined in [7], those parameters are used for understanding the quality of the KGs expressed in terms of

- *flexibility*, the property of an ontology to be easily adapted to multiple views;
- *transparency*, the property of an ontology to be analysed in detail, with a rich formalisation of conceptual choices and motivation;
- *cognitive ergonomics*, the property of an ontology to be easily understood, manipulated, and exploited by its consumers.

A wide range of metrics have been defined in the literature and can be computed with the help of dedicated tools, in particular OntoMetrics[27] and the more recent NEOntometrics[28]. In the following we report on the most representative metrics, also in relation to the aforementioned quality dimensions. Our analysis focuses on the reference domain-specific modules of the ontology networks for medical diagnostics and climate services, which rely on an import top-level and common modules (as well as any other domain specific module designed to represent a restricted knowledge area). The figures and numbers reported in the rest of this section thus refer to the core entry points to the ontology networks, representing the modules capturing the knowledge specific to our target domains of interest. This includes:

- the SNOMED Clinical Terms ontology module (HACID SNOMED Clinical Terms, HSCT), as core entry point to domain knowledge derived from the SNOMED CT resource;
- the Medical Diagnostics ontology module (MDX), as core entry point to domain knowledge related to the description of clinical cases and differential diagnoses made by healthcare professionals;
- the Climate Services ontology module (CCSO), as core entry point to domain knowledge for climate services.

Table 10 hereafter describes the base descriptive knowledge graph metrics used, along with their corresponding results as recorded for the HSCT, MDX and CCSO ontologies[29].

Table 10: Base knowledge graph metrics as computed for the HSCT, MDX and CCSO modules and related networks.

Metric	Description	HSCT	MDX	CCSO
# of axioms	The total number of axioms defined for classes, properties, datatype definitions, assertions and annotations.	1,573	1,375	1,545
# of logical axioms	The axioms which affect the logical meaning of the ontology network.	678	447	579
# of classes	The total number of classes defined in the ontology network.	127	52	79
# of object properties	The total number of object properties defined in the ontology network.	116	94	127
# of datatype properties	The total number of datatype properties defined in the ontology network.	26	10	15
# of annotation assertions	The total number of annotations in the ontology network.	641	767	727

[27] https://ontometrics.informatik.uni-rostock.de

[28] http://neontometrics.informatik.uni-rostock.de

[29] All metrics that consider the total number of axioms, classes, object properties etc take into account all modules that are imported from the reference ontology module. The term ‘ontology network’ in the metrics descriptions thus refers to the network of ontology modules reachable (as a result of imports) from the starting reference module.

Metric	Description	HSCT	MDX	CCSO
DL expressivity[30]	The Description Logics expressivity of the ontology network.	$\mathcal{SRI}(\mathcal{D})$[31]	$\mathcal{SRIQ}(\mathcal{D})$[32]	$\mathcal{SRIQ}(\mathcal{D})$
# of individuals	The total number of individuals in the knowledge graph.	4,517,399	1,044,647	173,218
# of triples	The total number of triples available in the knowledge graph.	27,427,383	5,030,445	1,573,946

Concerning the number of individuals and triples in the domain knowledge graph derived from SNOMED CT (i.e., HSCT in the Table), it is worth mentioning that a significant proportion of individuals (~80%) and triples originate from the representation of names for SNOMED concepts (i.e., fully specified names, preferred terms and alternative terms) according to the Naming ontology module. As detailed in D2.1, the Naming ontology module models the assignment of a name to an entity and allows representing entity names as first class citizens (going beyond simple datatype properties). The importance of the terminological perspective in SNOMED is thus reflected in the number of individuals and triples generated to represent concepts' names. This also becomes evident if we consider the class coverage for the HSCT knowledge graph, i.e., the number of individuals for each class. Table 11 shows the top-20 classes according to the number of individuals they have in the domain KG derived from SNOMED CT.

Table 11: Top-20 classes according to the number of individuals they have in the HSCT domain KG derived from SNOMED CT.

#	Class	# of individuals
1	`naming:Naming`	1,904,583
2	`naming:Name`	1,883,887
3	`hsct:DisorderDescription`	120,928
4	`hsct:ClinicalFinding`	105,966
5	`hsct:Disorder`	84,002
6	`hsct:ClinicalFindingDescription`	63,732
7	`hsct:Procedure`	48,212
8	`hsct:MedicinalProductDescription`	47,702
9	`hsct:ProcedureDescription`	44,384

[30] A reference to the labels used for representing DL expressivity (e.g. $\mathcal{SRI}(\mathcal{D})$, $\mathcal{SRIQ}(\mathcal{D})$, $\mathcal{SRIQ}(\mathcal{D})$, etc.) is available on Wikipedia at https://en.wikipedia.org/wiki/Description_logic, last visited on December 6th 2024.

[31] The $\mathcal{SRI}(\mathcal{D})$ DL expressivity is commonly used in ontologies that involve transitive properties, role hierarchies, inverse properties, and datatype reasoning. It balances expressivity and computational complexity.

[32] The $\mathcal{SRIQ}(\mathcal{D})$ is suitable for complex ontologies requiring cardinality constraints on specific relationships, like biomedical ontologies (e.g., restricting the number of diagnoses or symptoms associated with a patient).

#	Class	# of individuals
10	`hsct:AnatomicalStructure`	36,232
11	`hsct:Organism`	33,301
12	`hsct:EvaluationProcedureDescription`	27,489
13	`hsct:Substance`	27,363
14	`hsct:MedicinalProduct`	24,281
15	`hsct:SurgicalProcedure`	21,068
16	`hsct:EvaluationProcedure`	20,592
17	`hsct:AnatomicalStructureDescription`	20,019
18	`hsct:Device`	13,297
19	`hsct:ObservableEntity`	10,580
20	`hsct:SurgicalProcedureDescription`	10,526

Concerning the part of the KG that represents clinical cases and diagnostics attempts made by healthcare professionals (i.e., the description of clinical cases and differential diagnoses produced in the context of the HumanDX platform), the number of individuals and triples changes with the number of clinical cases that are included in the KG. The figures on the number of individuals and triples reported before in Table 10 reflect the dimension of the KG as of the time of this writing, with 1,622 clinical cases in the KG. Similar observations hold for class coverage, as shown in Table 12 with the ranking of classes according to the number of individuals they have in the KG.

Table 12: Ranking of classes according to the number of individuals they have in the MDX domain KG for medical diagnostics.

#	Class	# of individuals
1	`mdx:Diagnosis`	261,409
2	`mdx:ClinicalEvaluation`	261,409
3	`mdx:RankedClinicalEvaluation`	259,740
4	`mdx:DiagnosticAttempt`	65,501
5	`mdx:RankedClinicalEvaluationCollection`	65,501
6	`top:TemporalEntity`	49,408
7	`mdx:DisorderDescription`	32,801
8	`mdx:Disorder`	15,318
9	`naming:Naming`	11,073
10	`naming:Name`	11,073

#	Class	# of individuals
11	`mdx:HealthcareProfessional`	7,270
12	`top:Person`	1,622
13	`mdx:ClinicalCase`	1,622
14	`mdx:Finding`	763
15	`mdx:FindingSequence`	50
16	`mdx:Rank`	49
17	`mdx:Specialty`	38

Moving to the climate services domain, Table 13 shows the ranking of classes according to the number of individuals they have in the domain KG.

Table 13: Ranking of classes according to the number of individuals they have in the domain KG for climate services.

#	Class	# of individuals
1	`data:Dataset`	168,556
2	`data:Variable`	1,549
3	`ccso:Simulation`	1,482
4	`ccso:DynamicalDownscaling`	1,330
5	`data:TemporalRegion`	751
6	`ccso:GlobalSimulation`	152
7	`ccso:Model`	145
8	`top:Organization`	104
9	`ccso:ClimateModel`	87
10	`top:UnitOfMeasure`	66
11	`ccso:RegionalClimateModel`	46
12	`ccso:GlobalClimateModel`	41
13	`top:Activity`	24
14	`data:GeodeticRegion`	15
15	`gml:Rectangle`	15
16	`geo:Geometry`	15
17	`ccso:Realm`	14
18	`data:TimeDuration`	9

#	Class	# of individuals
19	ccso:GreenhouseGasConcentrationPathway	4

Additional schema and graph metrics for the three reference ontology modules and related networks are provided in Table 14. Each metric is related to the corresponding quality properties defined before (i.e, flexibility, transparency and cognitive ergonomics), according to the mapping defined in [7]. For each metric, the corresponding description is provided; additional details on the metrics can be found in the documentation available in the OntoMetrics Wiki[33].

[33] https://ontometrics.informatik.uni-rostock.de/wiki/index.php/OntoMetrics

Table 14: Schema and graph metrics as computed for the HSCT, MDX and CCSO modules and related networks.

Metric	Description	Quality property	HSCT	MDX	CCSO
Relationship richness	The ratio between non-inheritance relations and the total number of relations defined in the ontology. Inheritance relations are `rdfs:subClassOf` axioms. Values range from 0 (i.e., the ontology contains inheritance relationships only) to 1 (i.e. the ontology contains non-inheritance relationships only).	Transparency	0.26	0.49	0.56
Inheritance richness	The average number of subclasses per class. Inheritance Richness (IR) is expressed on an ordinal scale and its values should be interpreted relatively to the number of classes. If the number of subclasses is much smaller than the number of classes, then the value is low. On the contrary, if the number of subclasses tends to equalise the number of classes, the value is high. A low value indicates a deep (or vertical) ontology, while a high value indicates a shallow (or horizontal) ontology.	Transparency	2.54	1.9	1.32
Axiom/class ratio	The ratio between axioms and classes computed as the average amount of axioms per class. Its values should be interpreted relatively to the number of classes and axioms. Low values (i.e., close to 0) indicate poorly axiomatised ontologies. On the contrary, higher values indicate better axiomatisation. Extremely high values might indicate over axiomatisation.	Transparency	12.39	26.44	19.56
Class/property ratio	The ratio between the number of classes and the number of properties. Typically good values are in the range [0.3, 0.8] indicating a sufficient number of properties connecting things with other things (i.e., object properties) and values (datatype properties). Low values (i.e., close to 0) indicate an ontology with many properties connecting few concepts. On the contrary, high values indicate an ontology with many concepts connected by few properties. Nevertheless, the interpretation of the ratio always depends on the ontology size.	Cognitive ergonomics	0.29	0.27	0.33

Metric	Description	Quality property	HSCT	MDX	CCSO
Number of root classes (NoR)	The number of root classes. A root class is a class that is not a subclass of any other class in the ontology. NoR values are on ordinal scale and provide an indication of cohesion, i.e., the degree of relatedness between the different ontological entities. The interpretation of NoR values depends on the number of classes in the ontology. For example, 8 as NoR value might be low or high if the number of classes is 300 or 10, respectively.	Transparency, flexibility	6	3	3
Number of leaf classes (NoL)	The number of leaf classes. A leaf class is a class that has no subclass in the ontology. NoL values are on ordinal scale and provide an indicator of cohesion, i.e., the degree of relatedness between the different ontological entities. Again, the interpretation of NoL values depends on the number of classes in the ontology. For example, 8 as NoL value might be low or high if the number of classes is 300 or 10, respectively.	Transparency, flexibility	112	37	50
Average breadth	The average breadth computed on the graph whose nodes are ontology classes and edges are `rdfs:subClassOf` axioms. The metric suggests the degree of horizontal modelling (i.e., flatness) of the hierarchies of an ontology. Values are on an ordinal scale. The value should be interpreted relatively to the number of classes. For example, average breadth values of 10 and 100 in an ontology consisting of 600 classes are low and high, respectively.	Cognitive ergonomics	8.1	3.24	2.62
Max breadth	The maximal cardinality recorded on ordinal scale over the graph constructed as for the average breadth. The interpretation of max breadth is similar to that suggested for the average breadth.	Cognitive ergonomics	42	17	13
ADIT-LN	It records the average depth of the graph constructed as for the average breadth. The average is computed as the sum of the depth of all paths divided by the total number of paths. ADIT-LN values are on an ordinal scale and are indicators of cohesion. The interpretation of the values depends on the size of the ontology. Accordingly, low values occur when ADIT-LN is significantly lower than the number of classes. On the contrary, high values occur when the difference between ADIT-LN and the number of classes is less significant.	Transparency, cognitive ergonomics	2.59	2.8	3.95

Metric	Description	Quality property	HSCT	MDX	CCSO
Max depth	The maximal depth obtained by traversing `rdfs:subClassOf` axioms in the graph constructed as for the average breadth. The interpretation of max depth is similar to that suggested for ADIT-LN.	Cognitive ergonomics	3	5	10
Tangledness	The degree of multi-hierarchical nodes in the class hierarchy computed according to the formula provided in [7]. A multi-hierarchical node is a class having multiple super classes. Values for tangledness range from 0 (i.e., no tangledness) to 1 (i.e., each concept in the ontology has multiple super classes).	Cognitive ergonomics	0.39	0.46	0.20

While the metrics cannot provide precise conclusions or an exhaustive representation of the ontologies' characteristics, they offer indicative insights and reflect certain aspects of the ontology modules, as first summarised and then discussed in detail in the following.

- The **HSCT ontology** shows a predominantly hierarchical organization. It features rich semantic relationships with extensive connectivity between concepts, prioritizing relationships over introducing new concepts. The ontology demonstrates good cohesion through a small number of root classes, while maintaining detailed granularity at lower levels. Its moderate depth and breadth, along with some multi-hierarchical organization, creates a balance between detailed medical concept representation and practical usability. These characteristics align well with SNOMED CT's purpose as a clinical terminology system.
- The **MDX ontology** demonstrates a balanced approach between inheritance and non-inheritance relationships. The ontology shows rich semantic connections with a high axiom/class ratio, while maintaining a property-rich structure with a focused set of core concepts. It features high cohesion through few root classes and many detailed leaf classes, reflecting its specialized focus on medical diagnostics. The structure maintains moderate depth and breadth, with some horizontal expansion due to its general `Entity` class. The ontology's relatively shallow nesting and moderate multi-hierarchical connections create a balance between interconnectedness and usability, making it well-suited for representing medical diagnostic concepts and relationships.
- The **CCSO ontology** shows a balanced approach between hierarchical and associative relationships, with rich axiomatization supporting detailed connections between concepts. The ontology demonstrates strong cohesion through few root classes and numerous leaf classes, reflecting its specialized focus on climate services. While maintaining a moderately narrow and vertical structure overall, it features some horizontal expansion through top-level concepts like the `Entity` class. The ontology exhibits moderate depth with some areas of deeper specialization, particularly in domain-specific concepts like data structures. Its low tangledness and balanced property-to-class ratio create a structure that prioritizes both expressiveness and manageability, making it well-suited for capturing detailed climate service concepts and relationships.

HSCT ontology

For the HSCT ontology, the *relationship richness* suggests the ontology is relatively hierarchical, with limited diversity in the types of relationships represented. This reflects the taxonomic nature of SNOMED CT. Similarly, *inheritance richness* indicates a moderately deep class hierarchy, with a design prioritising hierarchy, specialisation and granularity over broad, horizontal structures. The vertical nature of the ontology suggests it covers a specific domain in a detailed manner, and this is in line with the goal of SNOMED as domain-specific resource for clinical concepts.

The *axiom/class ratio* suggests that the ontology has a rich set of axioms describing relationships, constraints, and semantics for the classes. This points to a detailed and expressive ontology. While the *class/property ratio* is slightly below the typical range, it indicates that the ontology has relatively more properties than classes, i.e., there is a rich set of properties connecting fewer concepts. This suggests a property-rich ontology with strong connectivity, in line with the nature of SNOMED, with a focus on relationships and attributes, emphasizing the connectivity and interactions between clinical concepts rather than introducing a large number of distinct concepts.

The *number of root classes* indicates a reasonably cohesive ontology, where the majority of the classes are hierarchically organised under a small set of root categories. This reflects a well-structured and logical design, allowing for clarity without excessive fragmentation. The relatively high *number of leaf classes* suggests an ontology that is highly detailed at the lower levels, with most classes being terminal. While this could affect cohesion by making the ontology less navigable or harder to generalize from, this should be interpreted in relation to the domain-specific nature of the ontology. Having an ontology that is highly detailed at the lower levels and aims at capturing specific, terminal concepts can be considered appropriate if it targets the medical domain, which requires fine-grained detail to accurately represent the specificities of disorders, clinical findings, procedures, etc.

In line with the discussion so far, the relatively low *average breadth* suggests a moderately structured ontology with a reasonable degree of horizontal modeling. It reflects a balance between flatness and depth, favoring clarity and manageability while avoiding excessive hierarchy at any single level. The moderately high *max breadth* indicates that the ontology has a few classes with a fairly large number of direct subclasses, suggesting areas of significant horizontal expansion. This reflects the presence in the hierarchy of a general *Entity* class that is then specialised by the domain-specific classes representing disorders, substances, organisms, etc. Similarly, a generic *Description* acts as the root class for all concept-specific descriptions (Disorder description, Substance description, Surgical procedure description, etc.).

The relatively low value for the *ADIT-LN* suggests good cohesion, with a structure that avoids excessive depth, favoring a more manageable and less complex hierarchy, as classes are likely clustered in a manageable number of levels. Similarly, the relatively low *max depth* value suggests that the ontology is not overly complex in terms of hierarchical depth, which can contribute to better cohesion and easier navigation.

Finally, the *tangledness* indicates moderate multi-hierarchical nodes, where some classes have multiple superclasses but the majority likely do not, offering flexibility while keeping the ontology's hierarchy relatively manageable.

MDX ontology

For the MDX ontology, the *relationship richness* indicates a well-balanced ontology, with almost equal use of inheritance and non-inheritance relationships. This suggests that the ontology uses inheritance to structure the hierarchy of classes, but it also includes a substantial number of non-inheritance relationships, which can provide richer semantic connections between the concepts. Non-inheritance relationships are in fact fundamental to capture the relationships between clinical cases, patients, diagnostic attempts, differential diagnoses, etc. The relatively low *inheritance richness* suggests a moderately deep ontology with a vertical structure, where classes are primarily organized into a few levels of abstraction. This generally indicates a more focused or specialized domain ontology, and this is in line with the goal of the MDX module, which captures and reflects the specificities of the medical diagnostics domain, and in particular as represented in the HumanDX platform.

The relatively high *axiom/class ratio* suggests that the ontology has a rich set of axioms describing the relationships, constraints, and characteristics of the classes, contributing to rich semantic coverage and providing a comprehensive representation of the target domain. As for the HSCT module, the *class/property ratio* is slightly below the typical range, suggesting a property-rich ontology with a smaller number of key concepts that are described in detail by various relationships. This reflects the scope of the MDX ontology, where a relatively small number of classes capture core domain concepts (clinical cases, patients, findings, diagnoses, healthcare professionals and a few more) and various properties establish relationships among these concepts.

The relatively low *number of root classes* indicates a moderately centralized structure with high cohesion, where the majority of the classes are linked through a few root classes. This suggests that the ontology is well-organized, focusing on fewer top-level concepts (also relying on the top-level and common modules) that are extended with more specific subclasses. Similarly, the relatively high *number of leaf classes* implies that the ontology has a detailed representation of terminal concepts and is more focused on specific concepts rather than on hierarchical relationships between them. This is in line with the scope of a domain-specific ontology like the MDX module where the majority of the classes are specialised or detailed concepts rather than high-level generalisations.

The relatively small *average breadth* confirms a relatively narrow hierarchy with relatively few subclasses per class. The ontology does not favor broad horizontal expansions, which is in line with the characteristics of the target domain where the already mentioned specific core concepts are not grouped under broad categories. Yet, the relatively high *max breadth* suggests the ontology has at least one broadly generalised class with a large number of direct subclasses, providing a significant horizontal expansion at that level. This is motivated by the presence of the general *Entity* class in the imported top-level module, which acts as a superclass for many, more specific classes in the hierarchy.

The relatively low value for the *ADIT-LN* suggests, as already observed, that the ontology does not have a deeply nested structure. The hierarchy is organized into a few levels and this appears appropriate for a domain where concepts do not require deep hierarchies and where relationships between entities can be represented without significant nesting. The *max depth* is moderate compared to the total number of classes and reflects a balanced depth, indicating again that the overall ontology avoids excessive verticality. This can support a mix of generalisation and specificity, which is often beneficial for maintaining clarity and usability, and thus good transparency and cognitive ergonomics.

Finally, the *tangledness* indicates a moderate level of multihierarchical connections, reflecting a balance between interconnectedness and navigability. While this could suggest that many classes belong to multiple hierarchies, the presence of property restrictions in class definitions should be taken into account as they contribute to induce multiple hierarchies.

CCSO ontology

For the **CCSO ontology**, the *relationship richness* reflects a balanced structure where both hierarchical and associative relationships are significant components. This suggests a structure designed for semantic richness and expressiveness, suitable for domains requiring detailed contextual and functional connections between entities, as it is the case of climate services. Domain specificity is also reflected in the relatively low *inheritance richness*, which indicates a moderately hierarchical and granular structure indicating a focus on capturing specific, fine-grained distinctions within the domain.

The relatively high *axiom/class ratio* suggests that the ontology is richly axiomatized, with a strong focus on capturing detailed relationships, constraints, and properties for each class. This makes the ontology tailored for advanced applications where such expressiveness is required. The *class/property ratio* falls within the reference range, leaning towards a property-rich design, indicating that the ontology focuses on defining relationships between entities rather than just classifying concepts, reflecting again a structure where properties are adequately used to connect entities.

The low *number of root classes* suggests that the ontology is well-organized and hierarchical, with concepts grouped under a small number of top-level categories. This is also a consequence of the adoption of the top-level module, resulting in a high degree of cohesion, as most classes in the ontology are derived from a small number of top-level concepts. The relatively high *number of leaf classes* indicates a design where much of the ontology's detail is concentrated at the lower levels. This appears suitable for a domain ontology, where capturing detailed or domain-specific entities with fine-grained modeling is a priority.

The relatively low *average breadth* reflects an ontology with a moderately narrow and vertical structure, favoring specialization and depth over horizontal modeling. This reinforces the notion of a domain-specific ontology where conceptual granularity is important without extensive horizontal expansion at any given hierarchical level. While the *max breadth* could reveal the presence of some asymmetry in the design with at least one part of the ontology that is horizontally structured, this is motivated again by the presence of top-level concepts (such as the already mentioned *Entity* class) that act as core root classes for the classification hierarchy.

The relatively low *ADIT-LN* suggests a moderately deep structure with a reasonable degree of vertical hierarchy. The ontology's depth is modest in relation to its size and this points to a structure where concepts are not excessively layered. The balance between specialization (depth) and navigability favours understandability as part of the cognitive ergonomics perspective. Yet, the relatively high max depth indicates that at least one part of the ontology is fairly deep, with multiple levels of subclass relationships. This highly detailed narrow specialisation can be observed for domain specific concepts such as the `data:PeriodicRegularGrid` class (also shown in Figure 8) part of the data-centric perspective for the specification of variables, dimensional spaces and datasets, presented in Section 2.2.4.1. The definition of a domain-specific concept hierarchy, combined with the hierarchical structure of top level concepts, induces a fairly deep class hierarchy.

To conclude, in terms of *tangledness*, the low value suggests that the ontology has a moderate level of multihierarchical nodes, indicating a balance between hierarchical structure and complexity. The overall structure remains relatively clear and manageable, making the ontology relatively flexible but not excessively tangled.

3. Bottom-up knowledge generation

The previous sections have detailed the foundational methodologies, data sources, and domain-specific knowledge graph implementations for both medical diagnostics and climate services. While these approaches primarily followed a top-down methodology through ontology design patterns and expert knowledge integration, the effectiveness of knowledge graphs can be significantly enhanced through bottom-up knowledge generation techniques. This complementary approach leverages the vast amounts of unstructured and semi-structured data available in both domains to automatically extract, validate, and integrate knowledge into the existing graph structures. By combining both top-down and bottom-up approaches, we can create more comprehensive and dynamic knowledge graphs that not only reflect expert-designed schemas but also capture emerging patterns and relationships from real-world data. The following section explores the methods and techniques employed for bottom-up knowledge generation, detailing how this approach enriches and validates our domain knowledge graphs. We have developed and evaluated the approach for the medical diagnostics use case, owing to the availability of annotated datasets. Nevertheless, the approach is readily applicable also to the climate service use case.

3.1. Structured Information Extraction in the Medical Domain

In this section, we describe the approach exploited within the HACID project for bottom-up knowledge generation,[34] previously introduced in the Deliverable D2.1 [1]. Among the several possibilities, we focus on structured information extraction from medical text, in order to enrich the KG with additional information retrieved from relevant text, and further support the evidence-based decision support developed by HACID. We first detail the goals of such a structured information extraction. We then proceed with detailing the tasks and the methodology employed, the selection of language and embedding models, and the devised prompt engineering methods. Building upon the conclusions presented in Deliverable D2.1 [1], our approach relies on recent advances in generative AI and Large Language Models (LLMs). This section focuses on the Retrieval Augmented Generation (RAG) paradigm for enhancing LLMs with contextual information [12]. RAG combines a pre-trained sequence-to-sequence model (the generator) with a neural retriever that accesses a dense vector index of relevant documents. The dense vector index is constructed by encoding a large corpus of text into high-dimensional vector representations using a dual-encoder architecture. Each document is embedded into a vector space where semantically similar texts are located closer to one another. The neural retriever efficiently searches this index to retrieve the most relevant documents for a given query, leveraging the semantic similarity between the query and the document vectors. By retrieving pertinent information based on the query, the model generates more accurate and contextually informed responses. The RAG framework improves the factual accuracy, adaptability, and scalability of language models, making them better suited for tasks requiring up-to-date or domain-specific knowledge. In our context the RAG approach is analysed for its accuracy, ensuring that the methodology aligns with the overarching goal of efficient knowledge graph construction.

[34] The Github repository for the RAG implementation and the experiments: hacid-project/hacid-RAG

3.1.1. Goals of structured information extraction

Given a medical text such as the title and abstract of a scientific article, we want to identify medical concepts and their relations, extracting triples `<subject,relation,object>` that (i) respect the ontology defined for the DKG and (ii) can be linked to concepts already existing in the DKG. Moreover, the extracted information is included into the DKG together with information about its provenance, following the pattern defined in Deliverable D2.1 [1].

3.1.2. Task Description

The task of knowledge graph construction is decomposed into the following sub-tasks, progressing from fundamental to complex levels.

Entity Extraction: To identify and extract meaningful entities from unstructured text, including all possible entities (e.g. both medical and non-medical entities). This step ensures comprehensive data collection to capture the full context of the input. For example, given the unstructured text: "*This was accounted for by a significant number of depressions occurring in methyl dopa treated patients with psychiatric histories."* The extracted entities could be ['significant number', 'depression', 'methyldopa', 'patient', 'psychiatric history'].

Entity-Type Linking: To associate only the desired entities with their corresponding predefined types (e.g. Disease, Chemical). This step narrows down the focus to domain-relevant concepts. Following the example above, here the output could be ('depression', 'Disease') ('methyldopa', 'Chemical') ('psychiatric history', 'Condition').

Triplet Extraction: To identify and structure relationships between medical entities into subject-predicate-object triplets, enriching the DKG with additional knowledge extracted from authoritative sources. Following the example above, here the output could be ('depression', 'has cause', 'methyldopa').

Description Generation: To generate a group of triplets that comprehensively describe an entity of interest. These triplets capture relationships between the entity and other concepts in the knowledge graph that make sense if taken together. Following the SNOMED-like structured format, in the example above the entity of interest "depression" could result in the following triples: ('depression', 'has cause', 'methyldopa'), ('depression', 'associated finding', 'psychiatric history'). In this way, the description captures the co-occurring relationships between concepts, which should not be neglected.

The sub-tasks described above are interlinked to form a cohesive and systematic process. **Entity Extraction** identifies all relevant data points from unstructured text, including medical and non-medical entities, laying the foundation for further processing. **Entity-Type Linking** narrows the focus to medically relevant entities by associating them with predefined types, ensuring domain specificity. These categorised entities are then structured through **Triplet Extraction**, which captures their relationships in the form of subject-predicate-object triplets, creating the backbone of the knowledge graph. Finally, **Description Generation** enriches the graph by producing descriptions where relevant, comprising groups of triplets that comprehensively define properties, associations, and roles of concepts within the input text. Together, these steps contribute to the generation of a robust, domain-specific knowledge graph that is both machine-readable and human-interpretable, enabling applications such as clinical decision support, diagnostics, and semantic search.

3.1.3. RAG structure

Two alternative configurations are explored to implement structured information extraction by means of a RAG approach. These configurations are depicted in Figure 12, and are referred to as **All-in-one RAG** and **Extractor+RAG** .

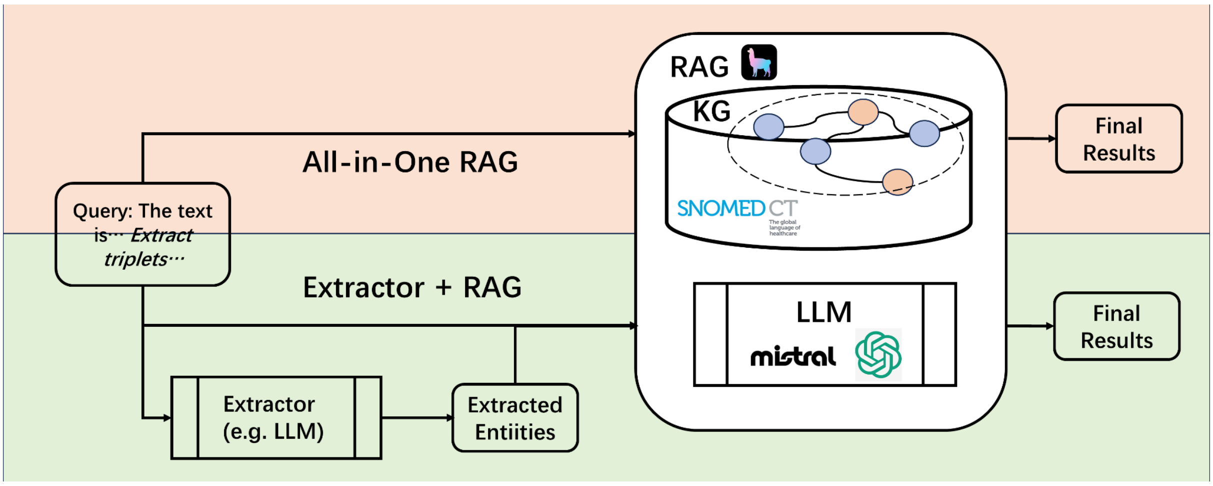


Figure 12. The workflows of All-in-One RAG and Extractor+RAG.

All-in-one RAG: In the All-in-one configuration, a single unified system performs all required sub-tasks in an end-to-end manner, as described in a single prompt. The devised workflow is detailed below.

- The unstructured text input is directly provided to the RAG model.
- The prompt explicitly describes the required sub-tasks in a sequential manner, outlining the expected outputs (e.g. triplets, and/or descriptions).
- The model processes the input holistically and outputs a fully structured set of entities, their types, triplets, or SNOMED-like descriptions in a single step.

Extractor+RAG: In this modular configuration, the workflow is divided into two stages. A dedicated extractor model handles the first subtask of Entity Extraction, and the RAG model focuses on downstream tasks (Entity-Type Linking, Triplet Extraction, and Description Generation). The rationale is to support the information extraction by first identifying all the possible entities, and then focus on the domain specific tasks. The workflow is as below:

- Extractor Stage:
  - The input text is provided to a specialised extractor model (e.g. an LLM, could be the same or different LLM to the one in RAG).
  - This model identifies all entities.
  - The output is a structured set of entities for RAG usage.
- RAG Stage:
  - The unstructured text input is fed directly to the RAG together with the extracted entities from the Extractor Stage, which provide an additional support.
  - The RAG system conducts the remaining sub-tasks using the extracted entities and relevant knowledge. For example, generating triplets representing relationships between entities, or producing SNOMED-like descriptions as groups of triplets for each entity.

The **All-in-one RAG** approach prioritises efficiency and simplicity, making it suitable for less complex or smaller-scale tasks. In contrast, the **Extractor+RAG** approach offers enhanced modularity and error control, making it ideal for large-scale or high-stakes applications where precision is critical. The choice between these configurations depends on the specific requirements and constraints of the knowledge graph construction task.

Knowledge Base: The knowledge base of RAG is built upon all the triplets extracted from the medical diagnostics knowledge graph derived from SNOMED CT (i.e., the KG that instantiates the HSCT ontology module). Specifically, the knowledge base for RAG was built by systematically querying the KG with a SPARQL query having the following core triple patterns.

```
SELECT ?fsn ?propertyLabel ?fsnOther
WHERE {

  [...]

  ?entity a ?entityClass ;
          hsct:isDescribedBy ?entityDesc ;
          hsct:fullySpecifiedName ?fsn .
  ?entityDesc a ?class ;
        ?property ?otherEntity .
  ?otherEntity a ?otherEntityClass ;
               hsct:fullySpecifiedName ?fsnOther .

  [...]

}
```

Basically, the query directly relates entities that in the KG are linked via a description. So if there is a path *entity → isDescribedBy → entityDescription → property → otherEntity*, a triplet involving *entity property otherEntity* is produced. Given the heterogeneity of concepts represented in SNOMED (and thus in the KG), the query constraints the source *entity* and the target *otherEntity* to be individuals of one of the following domain-relevant classes: Disorder, Clinical Finding, Substance, Organism, Morphologically Abnormal Structure. Fully specified names for SNOMED concepts and property labels are used to produce textual triplets as in the following example:

> Post-acute COVID-19 (disorder), causative agent, Severe acute respiratory syndrome coronavirus 2 (organism)

The resulting knowledge base consists of 236,547 triplets, with 23 different relations (e.g., causative agent, finding site, pathological process, has active ingredient, etc.) linking concepts of the aforementioned classes.

3.1.4. LLM and Embedding model selection

The selection of models plays a crucial role in optimising task-specific performance. Below are the shortlisted models:

- LLM generator

    - Mistral-Small-Instruct-2409[35] (***mistral***): A compact and efficient LLM designed for instruction-based tasks.
    - GPT-4o-mini[36]: A lightweight variant of GPT-4 optimised for resource constrained environments.
- Embedding Models
    - all-MiniLM-L12-v2[37] (***minilm12***): General-purpose embeddings suitable for a wide range of NLP tasks.
    - HiT-MiniLM-L12-SnomedCT[38] (***hitsnomed***): Domain-specific embeddings tailored for the SNOMED CT medical terminology.
    - Bge-m3[39] (***bgem3***): A versatile embedding model emphasising multi-modal compatibility.

3.1.5. Prompt Engineering

Prompt engineering ensures the effective communication of task requirements to LLMs. The process involves setting the context, providing necessary background knowledge and input data, clearly outlining the specific tasks to be performed, and providing additional requirements, such as specifying constraints, preferences, or desired formats for the output. Each aspect of prompt engineering is fine-tuned to enhance the accuracy and relevance of the generated results, contributing to the overall quality of the knowledge graph. We resorted to an ad-hoc methodology for prompt design, following best practices available in the community and testing/fine-tuning each part individually. The prompt design can be broken down into the four aspects:

1. **Introduction**:
    - The prompt begins with "{text}", indicating the context that is provided for processing
    - Sets the initial scope of working with entity-type extraction from a given context
    - Establishes that the task involves identifying entities and classifying them into specific types

```
Here is the context: {text}.\
Task: Extract the entity-type pairs from the given context with the format
of (entity ; type).\
```

2. **Information**:
    - Provides additional information necessary to perform the required tasks
        - For the task of entity-type linking, provides a clear set of constraints on the types of entities that can be identified

```
Here is the type list: [Disorder, Substance].\
```

        - For the extractor+RAG, an additional extracted entities list will be provided following the type list

```
Here is the type list: [Disorder, Substance].\
```

[35] mistralai/Mistral-Small-Instruct-2409 · Hugging Face

[36] GPT-4o mini: advancing cost-efficient intelligence | OpenAI

[37] sentence-transformers/all-MiniLM-L12-v2 · Hugging Face

[38] Hierarchy-Transformers/HiT-MiniLM-L12-SnomedCT · Hugging Face

[39] BAAI/bge-m3 · Hugging Face

```
Here is the list of entities for consideration: {entities}.\
```

   - For the task of triple extraction and description generation, the relation list will be provided instead of type list

```
Here is the relation list: [temporally follows, after, due to,
has realization, associated with, has definitional
manifestation, associated finding, associated aetiologic
finding, interprets, associated morphology, causative agent,
course, finding site, temporally related to, pathological
process, direct morphology, is modification of, measures,
direct substance, has active ingredient, using, part of].\
```

3. **Task Description**:
   - Detailed step-by-step process for task execution (e.g., entity-type pair extraction):
   - Suggests the use of a retrieved sub-graph as a method for entity extraction
   - Emphasises a systematic, algorithmic approach to task execution, depending on the task type
     - Task description component for entity-type linking

```
The steps are as follows:\
1. extract the entity from the given context abstract, using
the retrieved sub-graph.
2. select ONE most likely type from the list for the extracted
entity.
3. output the pairs in the format of (entity ; type) strictly.
4. repeat the step 1 to step 3.\
```

     - Task description component for triple extraction

```
The steps are as follows:\
1. extract the concept 1 and concept 2 from the given context
sentence, using the retrieved sub-graph.
2. select ONE most likely relation from the list for the
extracted concepts.
3. output the triples in the format of (concept 1 ; relation ;
concept 2) strictly.\
```

     - Task description component for description generation

```
The steps are as follows:
1. extract the concept 1 from the given context sentence,
using the retrieved sub-graph.
2. generate the concept 2 that can describe the concept 1, and
select ONE most likely relation from the list for the concept
1.
3. output (concept 1 ; relation ; concept 2) strictly as one
generated description.
4. Each extracted concept could have multiple descriptions.\
```

4. **Additional Requirements:**
   - Strict output format
   - Mandatory requirements
   - Specifies a disciplined approach to output presentation and type selection

```
Provide your answer as follows:
Answer:::
Pairs: (All extracted pairs)\
Answer End:::\

Requirements:\
You MUST provide values for 'Pairs:' in your answer. \
ONLY use the type in the type list: [Disorder, Substance].\
Extract as many valid entity-type pairs as possible from the given context
abstract.\
```

In the following, we provide full examples of prompts used for entity-pair extraction with All-in-One RAG, as detailed below.

```
Here is the context: {text}.\
Task: Extract the entity-type pairs from the given context with the format
of (entity ; type).\
Here is the type list: [Disorder, Substance].\

The steps are as follows:\
1. extract the entity from the given context abstract, using the retrieved
sub-graph.
2. select ONE most likely type from the list for the extracted entity.
3. output the pairs in the format of (entity ; type) strictly.
4. repeat the step 1 to step 3.\

Provide your answer as follows:
Answer:::
Pairs: (All extracted pairs)\
Answer End:::\

Requirements:\
You MUST provide values for 'Pairs:' in your answer. \
ONLY use the type in the type list: [Disorder, Substance].\
Extract as many valid entity-type pairs as possible from the given context
abstract.\
```

For entity-pair extraction with the extractor+RAG approach, the process is divided in two steps. First, the extraction of entities is performed with a simple prompt for the extractor:

```
Extract all entities from the following text: {text}.
ONLY respond with the ENTITIES without any reasoning.
Entities: []
```

Then, the prompt for RAG is built with a similar structure as described above, with the addition that the extracted entities. An example of a complete RAG prompt for the extractor+RAG approach is given below.

```
Here is the context: {text}.\
Task: link the entity and the type and output entity-type pairs with the
format of (entity ; type).\
Here is the type list: [Disorder, Substance].\
Here is the list of entities for consideration: {entities}.\

The steps are as follows:
1. for each entity in {entities}, link it to the most likely type from the
type list. if you cannot find a suitable type, ignore the entity.
2. if you find more entities in the abstract, extract them and link them to
the most likely type.
3. output the pairs in the format of (entity ; type) strictly.\

Provide your answer as follows:
Answer:::
Pairs: (entity ; type)
Answer End:::\

Requirements:
You MUST provide values for 'Pairs:' in your answer.
ONLY use the type in the type list: [Disorder, Substance].
ONLY output valid entity-type pairs without any reasoning.
```

3.2. Evaluation

This section outlines the datasets, metrics, experimental setups, and results used to evaluate the performance of the RAG-based knowledge extraction process. The goal is to assess the effectiveness of different configurations and parameter settings in achieving accurate and meaningful outputs across the sub-tasks. The evaluation has been conducted for the medical diagnostics case study, owing to the availability of several datasets that can be used for quantitative evaluation. In the future, the RAG-based knowledge extraction will be also used in the case study about climate change adaptation management.

3.2.1. Dataset

We have exploited diverse datasets to comprehensively test the entity-pair extraction, triplet extraction, and description generation tasks under different configurations. Following, we detail the dataset used, together with the objective stated for their usage, that is, either evaluation or exploitation. For evaluation, we aim at a quantitative appraisal of the quality of the proposed approach for one or more sub-tasks. For exploitation, we aim at producing new knowledge to be included in the DKG.

BC5CDR (evaluation)

- A benchmark biomedical dataset comprising 50 abstracts with gold-standard annotations for two entity types: disease and chemical.
- Used for evaluating the **entity-pair extraction task**, where the focus is on identifying entity types with high accuracy.

MIMIC-IV (evaluation)

- A large-scale, de-identified clinical dataset containing rich medical text data.

- Particularly, the SNOMED CT Entity Linking Challenge[40][41] is used, which is a subset of MIMIC-IV-Note discharge summaries that have been annotated with SNOMED CT concepts.
- The annotated dataset comprises 204 documents containing 51,574 annotations in which 5,336 distinct concepts appear.
- Used for evaluating **entity-pair extraction**, particularly in a clinical setting where the complexity and variability of language are high.

Pubmed abstracts (exploitation)

- A collection of unstructured abstracts and titles from PubMed was used for **triplet extraction** and **description generation** as the new knowledge in the expansion.
- We collected 24.73 million abstracts and titles up to date of 2023, categorised by year.

3.2.2. Metrics

Quantitative evaluation is performed for the entity-type linking task exploiting the BC5CDR and the MIMIC-IV dataset, which are both annotated with relevant entities. For these datasets, we compute standard metrics, such as:

- **Precision:** the proportion of correctly identified results among all retrieved results.
- **Recall:** the proportion of correctly identified results among all relevant results.
- **F1 Score:** harmonic mean of precision and recall

For exploitation, datasets do not come with annotated ground truth (e.g., PubMed Abstracts). An evaluation is anyway performed with the following approach:

- **LLM Judging:** LLMs are employed to rate the results on predefined criteria such as relevance and completeness.

3.2.3. Experiments

A series of experiments are conducted to compare configurations and parameter settings across multiple dimensions on different datasets. In this section, the experimental designs and the results are presented across the evaluation datasets

3.2.3.1. Experiments on BC5CDR

The goal of the experiments on BC5CDR is to evaluate the performance of the RAG system through extracting entities and linking them with the correct types. Here we present some experimental configurations to achieve our goals:

Parameters

- **Retrieved Top-K:** Determines the number of relevant context items retrieved for each query from the knowledge graph.

Comparison focuses

- All-in-one RAG vs. extractor+RAG: Compares the performance, flexibility, and accuracy of the two RAG configurations.
- High Top-K vs. low Top-K: Examines how varying the number of retrieved contexts affects the quality of outputs.

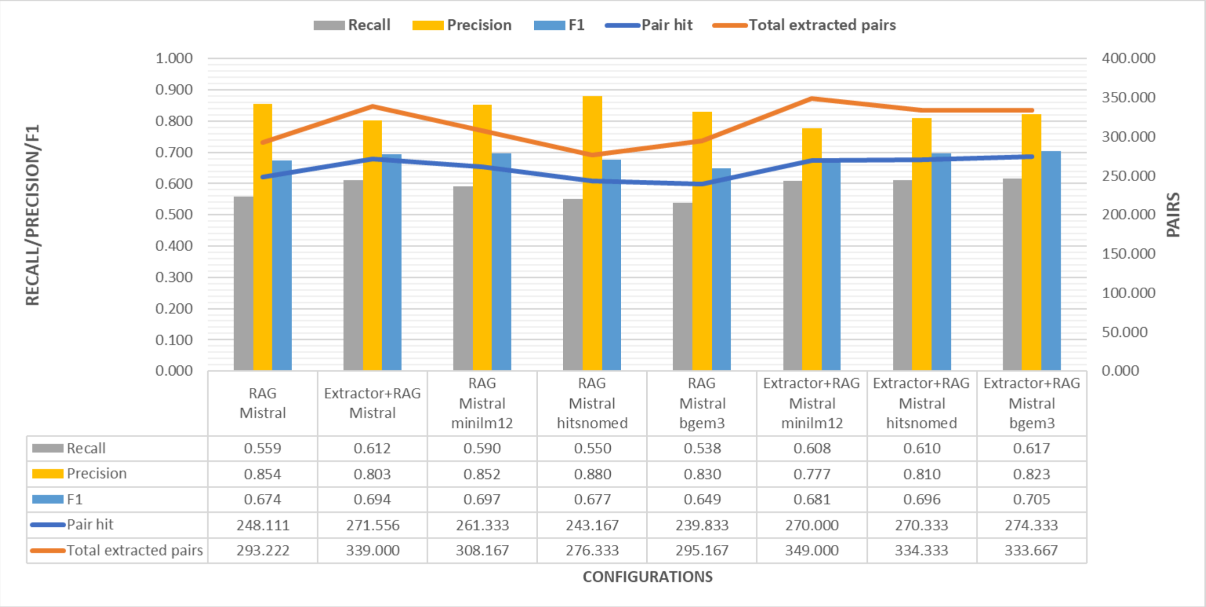
[40] Hardman, W., Banks, M., Davidson, R., Truran, D., Ayuningtyas, N. W., Ngo, H., Johnson, A., and Pollard, T. (2023) 'SNOMED CT Entity Linking Challenge' (version 1.0.0), PhysioNet. Available at: https://doi.org/10.13026/s48e-sp45.

[41] SNOMED CT Entity Linking Challenge v1.0.0

- Selections and combinations of LLM and embedding models: Evaluates different LLMs and embedding models (e.g., Mistral-Small + HiT-MiniLM-L12-SnomedCT) individually and in combination to identify the optimal setup.

Results

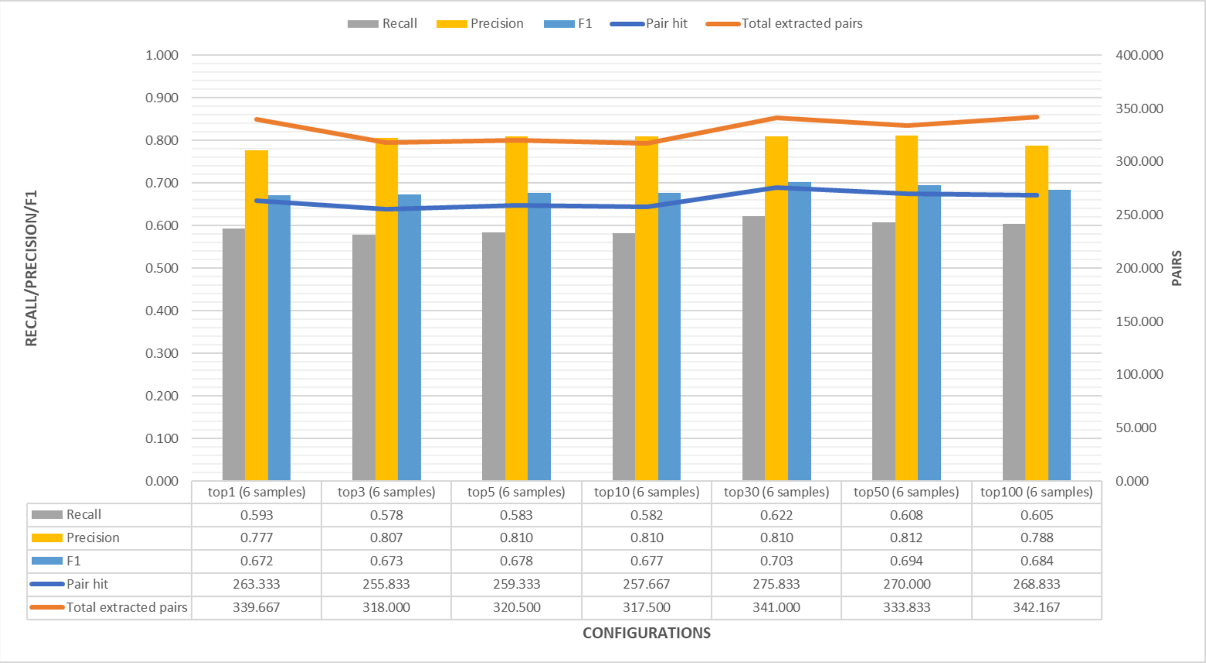
We first run evaluations on BC5CDR golden eval set, including 50 abstracts and annotated diseases and chemicals in each abstract. Figure 13 shows the comparative analysis that evaluates the All-in-One RAG configurations against Extractor+RAG approaches. The All-in-One RAG model (first column in Figure 13) demonstrates lower performance compared to the Extractor+RAG model (second column in Figure 13) for what concerns the F1 score. The All-in-One RAG has a higher precision, but recall is poor. This is also due to the lower amount of extracted pairs of All-in-One RAG (293) compared to Extractor+RAG (339).



	RAG Mistral	Extractor+RAG Mistral	RAG Mistral minilm12	RAG Mistral hitsnomed	RAG Mistral bgem3	Extractor+RAG Mistral minilm12	Extractor+RAG Mistral hitsnomed	Extractor+RAG Mistral bgem3
Recall	0.559	0.612	0.590	0.550	0.538	0.608	0.610	0.617
Precision	0.854	0.803	0.852	0.880	0.830	0.777	0.810	0.823
F1	0.674	0.694	0.697	0.677	0.649	0.681	0.696	0.705
Pair hit	248.111	271.556	261.333	243.167	239.833	270.000	270.333	274.333
Total extracted pairs	293.222	339.000	308.167	276.333	295.167	349.000	334.333	333.667

Figure 13. The performance comparison between All-in-One RAG and Extractor+RAG on BC5CDR.

An additional comparative analysis of Mistral LLM configurations with different embedding models reveals nuanced performance variations. Extractor+RAG configurations consistently demonstrated recall above 0.6, maintained decent precision between 0.77-0.82, and achieved F1 scores slightly below or above 0.7. In contrast, the RAG configurations exhibited more extreme performance characteristics, with recall consistently under 0.6, precision ranging from 0.83-0.88, and F1 scores between 0.649-0.697. These results indicate that the Extractor+RAG approach provides more balanced and robust performance across recall, precision, and F1 metrics, suggesting its potential advantages in information extraction tasks compared to the monolithic All-in-One RAG model. Among all configurations, the extractor+RAG with the combination of *mistral* and *begm3* achieves the highest F1 score (0.705) along with the highest recall (0.617).



	top1 (6 samples)	top3 (6 samples)	top5 (6 samples)	top10 (6 samples)	top30 (6 samples)	top50 (6 samples)	top100 (6 samples)
Recall	0.593	0.578	0.583	0.582	0.622	0.608	0.605
Precision	0.777	0.807	0.810	0.810	0.810	0.812	0.788
F1	0.672	0.673	0.678	0.677	0.703	0.694	0.684
Pair hit	263.333	255.833	259.333	257.667	275.833	270.000	268.833
Total extracted pairs	339.667	318.000	320.500	317.500	341.000	333.833	342.167

Figure 14. The performance between high Top-K and low Top-K configurations on BC5CDR, evaluated with the Extractor+RAG approach (Mistral).

Figure 14 compares the performance of an Extractor+RAG (Mistral) model across different Top-K settings, focusing on the contrast between low Top-K (1, 3, 5, 10) and high Top-K (30, 50, 100). At low Top-K, the recall, precision, and F1 scores are lower, ranging from 0.672 to 0.678. Pair hits and total extracted pairs are also lower, between 263,333 and 259,333.

In contrast, the high Top-K settings show higher performance, with the recall of 0.605-0.622, the precision of 0.788–0.810, and F1 of 0.684–0.703. Pair hits increase to 275,833–291,000, and total extracted pairs reach 341,000–376,000. Within the group of high Top-K, however, performance slightly decreases with increasing number of retrieved items. The tradeoff is that higher Top-K extracts more candidate pairs, potentially requiring more computational resources and possibly making more mistakes with irrelevant context. The best conditions for recall, precision, and F1 suggest the best configuration for this Extractor+RAG task is Top-30.

3.2.3.2. Experiments on MIMIC-IV

Similarly as the experiments on BC5CDR, we further conduct an experiment on the MIMIC-IV challenge dataset, which consists of 204 discharge notes and 51,574 annotated SNOMED-CT concepts (5,336 distinct concepts). Each concept was assigned a type by referencing our knowledge graph using SNOMED IDs, creating paired annotations of discharge notes and (concept, type) pairs.

Due to the extensive length of discharge notes compared to paper abstracts, we segmented each note into chunks of 1,000 tokens, resulting in 2,055 discharge note chunks with their corresponding concept-type pairs. For computational efficiency, we selected 150 discharge notes and their associated pairs for our experimental evaluation.

The configuration of the experiments are as below:

- Implementation focused on the extractor+RAG configuration
- RAG system utilized two LLMs: Mistral and GPT-4o-mini

- Two embedding models were tested: BGEM3 and HitsnoMed
- Mistral LLM served as the individual extractor
- Various Top-K values (30, 50, 100) were evaluated

Results

We present the results from two levels, pair level and entity level. For the pair level, the extracted entity and its identified type will be compared with the annotation. While for the entity level, only the entities are checked with the annotation.

Table 15: Results of the experiments on MIMIC-IV on the pair level.

Configurations	Recall	Precision	F1
Mistral+bgem3	0.186	0.250	0.213
Mistral+hitsnomed	0.198	0.270	0.228
gpt-4o-mini+bgem3	0.177	0.269	0.213
gpt-4o-mini+hitsnomed	0.195	0.289	0.232

Table 15 shows the results of the experiment on the pair level. The pair-level evaluation assessed both entity extraction and type identification accuracy. The results demonstrate that: The Mistral+hitsnomed configuration achieved the highest recall (0.198) and F1 score (0.228). GPT-4o-mini+hitsnomed showed superior precision (0.289) among all configurations. All configurations maintained relatively consistent performance, with F1 scores ranging from 0.213 to 0.232.

Table 16: Results of the experiments on MIMIC-IV on the entity level.

Configurations	Recall	Precision	F1
Mistral+bgem3	0.389	0.532	0.449
Mistral+hitsnomed	0.402	0.551	0.464
gpt-4o-mini+bgem3	0.379	0.577	0.458
gpt-4o-mini+hitsnomed	0.396	0.587	0.473

Table 16 shows the results of the experiment on the entity level. The entity-level evaluation focused solely on entity extraction accuracy, showing notably higher performance metrics. The Mistral+hitsnomed configuration demonstrated the best recall (0.402) and F1 score (0.464), while the GPT-4o-mini+hitsnomed configuration achieved the highest precision (0.587). The performance gap between different configurations was minimal, with F1 scores ranging from 0.449 to 0.473.

The substantial difference between pair-level and entity-level metrics suggests that while entity extraction is relatively robust (with F1 scores ranging from 0.449 to 0.473), accurate type identification remains challenging (with significantly lower F1 scores of 0.212 to 0.232). This performance gap indicates potential areas for improvement in the type assignment mechanism of the RAG system, primarily due to two key challenges.

First, the mapping between SNOMED concepts and their types introduces an additional layer of complexity. This is evident in the sharp drop in precision from entity-level (0.532-0.594) to pair-level performance (0.253-0.291), suggesting that even when entities are correctly identified, the system struggles with accurate type assignment.

Second, medical concepts often have multiple potential classifications, making precise type determination challenging. This complexity is reflected in the consistently lower recall rates at the pair level (0.175-0.196) compared to entity level (0.377-0.400), indicating that the system has difficulty selecting the correct type from multiple possible classifications.

These challenges point to the need for more sophisticated type assignment mechanisms that can better handle the inherent complexity of medical concept classification while maintaining the robust performance achieved in entity extraction.

Case study

Looking at the raw number of recall, accuracy and F1 provides a dry quantitative evaluation of performance of the selected approach. However, to better appreciate how the proposed RAG system behaves with respect to the information extraction task, it is also useful to look at the generated knowledge and evaluate what errors are performed in relation to the annotated dataset. We provide in the following two selected cases, one in which the quantitative evaluation of the RAG system was positive, and one in which it was instead negative.

A case with good results

In the case presented below, the RAG information extraction presents a good performance (F1 = 0.731). We show here the discharge note chunk, and a table with annotated concept-type pairs and extracted pairs for cross reference. The concepts are marked in different colours in both the free text and the concept table, representing whether the extracted entities or types are correctly identified (true positives in green), not recognised (false negatives in red), or wrongly recognised (false positives in purple).

Discharge Note chunk: *Touching her skin exacerbates the pain. She reports that even when sleeping, when her sheets touch the ___ skin, it wakes her from sleep. She is unable to wear underpants or pants ___ to pain. She has never experienced this before; she recovered well after her liposuction procedure. + nausea when pain is worst, denies emesis. Tolerating liquids, pudding, and yogurt. Passing flatus. Denies fevers, chills, abnormal vaginal discharge or bleeding. Has had occasional hot flashes and vaginal dryness. Not sexually active. In the ED, she received morphine 8mg IV, zofran 4mg IV, and dilaudid 0.5 IV. The dilaudid has had the best effect. She had 2L of NS. Past Medical History: GYNHx: - denies h/o abnl pap, last pap ___ neg - Denies h/o STI - female partners ___: GO PMH: Mild asthma, chronic back pain - disc degeneration, GERD, Depression, Insomnia PSH: - TAH BSO as above - Liposuction x 2, ___ - stomach and thighs Social History: ___ Family History: NC*

Table 17: Annotations and extracted entity-type pairs for the case with good performance.

Annotated Ground-truth concept-type pairs	Annotated SCTID and Primary concept	Extracted concept-type pairs (Hit / Miss / Wrong)
(pain ; finding)	22253000, Pain	(pain ; finding)
(skin ; body structure)	39937001, Skin structure	(skin ; body structure)
(sleeping ; finding)	248220008, Asleep	
(liposuction procedure ; procedure)	302441008, Liposuction of subcutaneous tissue	(liposuction ; procedure)
(nausea ; finding)	422587007, Nausea	(nausea ; finding)
(emesis ; disorder)	422400008, Vomiting	(emesis ; finding)
(Tolerating liquids ; finding)	473358000, Tolerating oral fluid	(liquids ; regime/therapy)
(Passing flatus ; finding)	249504006, Tolerating oral fluid	(flatus ; finding)
(fevers ; finding)	386661006, Fever	(fevers ; finding)
(chills ; finding)	43724002, Chill	(chills ; finding)
(abnormal vaginal discharge ; finding)	289567003, Vaginal discharge problem	(vaginal discharge ; finding)
(bleeding ; finding)	289530006, Bleeding from vagina	(bleeding ; finding)
(vaginal dryness ; disorder)	31908003, Vaginal dryness	(vaginal dryness ; finding)
(Not sexually active ; finding)	162171002, Currently not sexually active	
(IV ; regime/therapy)	386340006, Intravenous therapy	
(abnl pap ; finding)	309081009, Abnormal cervical smear	
(STI ; disorder)	8098009, Sexually transmitted infectious disease	
(Mild asthma ; disorder)	370218001, Mild asthma	(asthma ; disorder)
(chronic back pain ; finding)	134407002, Chronic back pain	(back pain ; disorder) (pain ; finding)
(disc degeneration ; disorder)	77547008, Degeneration of intervertebral disc	(disc degeneration ; disorder)
(GERD ; disorder)	235595009, Gastroesophageal reflux disease	(GERD ; disorder)
(Depression ; disorder)	35489007, Depressive disorder	(depression ; disorder)
(Insomnia ; disorder)	193462001, Insomnia	(insomnia ; disorder)

Annotated Ground-truth concept-type pairs	Annotated SCTID and Primary concept	Extracted concept-type pairs (Hit / Miss / Wrong)
(TAH BSO ; proce-dure)	**116144002, Total abdominal hysterectomy with bilateral salpingo-oophorectomy**	(TAH BSO ; procedure)
(Liposuction ; procedure)	**302441008, Liposuction of subcutaneous tissue**	(liposuction ; procedure)
(stomach ; body structure)	**69695003, Stomach structure**	(stomach ; body structure)
(thighs ; body structure)	**61396006, Structure of left thigh**	(thighs ; body structure)
		(pudding ; regime/therapy)
		(yogurt ; regime/therapy)
		(hot flashes ; finding)
		(family history ; finding)

As shown in Table 17, in this case, the discharge note chunk contains 27 ground-truth pairs, and our approach extracts 25 pairs, within which 19 pairs match with the ground-truth. Therefore, we have the following evaluation results:

- Precision: 19 / 25 = 0.76
- Recall: 19 / 27 = 0.704
- F1: 2 * precision * recall / precision + recall = 0.731

From the entity aspect, all extracted entities appeared in the free text chunk faithfully. Within them, considering the wrong pairs, pudding (SCTID: 711611001) and yogurt (SCTID: 226863004) can be found in the SNOMED dataset in the "substance" type class. Hot flashes (SCTID: 198436008) can also be found in SNOMED with "finding" class type, which indicates that our approach extracted an accurate unannotated pair. The other wrong pairs are with the situation where wrong types are linked, particularly the finding/disorder confusion, of which the boundary is not absolutely clear, to the point that the annotation itself could be questioned. For example, emesis is normally treated as a finding rather than a disorder.

For those annotated pairs that the RAG misses, the most common situation is that they are abbreviations (e.g. abnl pap) or acronyms (e.g. IV, STI), which increases the extraction difficulty.

A case with bad results

The case presented below corresponds to quite bad performance (F1 = 0.063). In the following, we analyse in detail the results to better understand how such a low score has been obtained.

Discharge Note chunk: *Name: ___ Unit No: ___ Admission Date: ___ Discharge Date: ___ Date of Birth: ___ Sex: F Service: MEDICINE Allergies: No Known Allergies / Adverse Drug Reactions Attending: ___. Chief Complaint: presyncope (feeling faint, acute vision changes, palpitations, tightness in his chest)*

Major Surgical or Invasive Procedure: none History of Present Illness: ___ F h/o presyncopal episodes the in the last 2 days, associated with dyspnea as well as diaphoresis. Patient also reports increased shortness of breath on exertion. Two days prior to admission, the patient was standing in the bathroom (no full bladder, not moving bowels) when she experienced narrowing of her visual fields, disequilibium (not vertiginous without nausea) sudden in onset. This was followed by a tightening in the throat, diaphoresis, and heart palpitations. This resolved over the course of ___ minutes after she sat down. There was no hearing

Table 18: Annotations and extracted entity-type pairs for the case with bad performance.

Annotated Ground-truth concept-type pairs	Annotated SCTID and Primary concept	Extracted concept-type pairs (Hit / Miss / Wrong)
(Adverse Drug Reactions ; finding)	**419511003, Propensity to adverse reactions to drug**	
(presyncope ; disorder)	**427461000, Near syncope**	(presyncope ; disorder)
(feeling faint ; finding)	**248223005, Feeling faint**	(feeling faint ; disorder)
(palpitations ; finding)	**80313002, Palpitations**	(palpitations ; disorder)
(chest ; body structure)	**51185008, Thoracic structure**	(tightness in his chest ; disorder)
(dyspnea ; finding)	**267036007, Dyspnea**	(dyspnea ; disorder)
(diaphoresis ; finding)	**52613005, Excessive sweating**	(diaphoresis ; disorder)
(shortness of breath on exertion ; finding)	**60845006, Dyspnea on exertion**	(shortness of breath ; disorder)
(standing ; finding)	**10904000, Orthostatic body position**	
(bladder ; body structure)	**89837001, Urinary bladder structure**	
(moving bowels ; finding)	**300373008, Finding of defecation**	
(narrowing of her visual fields ; finding)	**267628004, Generalized visual field constriction**	(narrowing of her visual fields ; disorder)
(disequilibium[42] ; finding)*	**249990003, Unsteady when standing**	(disequilibium ; disorder)
(nausea ; finding)	**422587007, Nausea**	
(throat ; body structure)	**49928004, Structure of anterior portion of neck**	(tightening in the throat ; disorder)

[42] An original typo, should be "disequilibrium".

Annotated Ground-truth concept-type pairs	Annotated SCTID and Primary concept	Extracted concept-type pairs (Hit / Miss / Wrong)
(heart ; body structure)	**80891009, Heart structure**	(heart palpaitations ; disorder)
(palpaitations ; finding)*	**80313002, Palpitations**	(heart palpaitations ; disorder)
(shortness of breath ; finding)	**267036007, Dyspnea**	(shortness of breath ; disorder)
(syncope ; finding)	**271594007, Syncope**	(presyncope ; disorder)
		(acute vision changes ; disorder)
		(presyncopal episodes ; disorder)

As shown in Table 18, in this case, the discharge note chunk corresponds to 19 ground-truth pairs, while our approach extracts 13 pairs, within which only one pair matches with the ground truth. Therefore, we have the following evaluation results:

- Precision: 1 / 13 = 0.077
- Recall: 1 / 19 = 0.053
- F1: 2 * precision * recall / precision + recall = 0.063

Looking at the details, most of the wrong pairs suffer from the confusion between finding and disorder, despite the entities all faithfully appearing in the original text. Another situation happens when a body structure is specifying a finding, which is recognised as a single entity by the RAG system (i.e., heart palpitations[43], tightening in the throat), but in the text it is annotated only with the body structure (i.e., heart, throat). This issue could be solved if nested entities would be annotated and extracted, but this is not considered either in the dataset or in the information extraction prompt.

Overall, we can conclude that, beyond the quantitative evaluation, the RAG system could extract relevant information from the given text, although sometimes with difficulty in assigning the correct type to the extracted concepts. While we could work to improve this aspect (e.g., working on the type of triples extracted from the KG as well as on the length of the chunks to be processed), we believe that the RAG system should be evaluated for the overall information extraction capabilities—our ultimate goal—and not for the entity-type linking task, which is not always necessary and could be addressed through a correct matching with the knowledge graph (see also Section 3.3). To this end, we present in the following section a set of experiments performed on Pubmed abstracts.

3.2.3.3. Experiments on Pubmed

The experiments on Pubmed aim to extract valid triples for KG enrichment using RAG-based approaches. The experimental dataset comprises 50 randomly selected abstracts from PubMed publications that appeared in 2023. This sampling approach was designed to ensure: 1) Temporal relevance through the use of recent (2023) publications. 2) Diverse biomedical subject coverage from PubMed's comprehensive database. 3) Manageable

[43] A typo in the original discharge notes, should be “palpitations”,

evaluation scope while maintaining statistical significance. The configuration of the experiments are as below:

- We test extractor+RAG with the LLM *mistral* and embedding model *bgem3*, following the best configuration presented in Figure 13.
- The individual extractor is set by LLM *mistral* as well.
- We test different Top-K values: 30, 50, and 100

The prompt for extracting triples from this piece of text as below

```
Here is the context: {text}.\

Task: Extract the SNOMED CT triples from the given context with the format
of (concept 1 ; relation ; concept 2).\

Here is the optional relation list: [temporally follows, after, due to, has
realization, associated with, has definitional manifestation, associated
finding, associated etiologic finding, interprets, associated morphology,
causative agent, course, finding site, temporally related to, pathological
process, direct morphology, is modification of, measures, direct substance,
has active ingredient, using, part of].\

The steps are as follows:\
1. extract the concept 1 and concept 2 from the given context sentence,
using the retrieved triplets.
2. select ONE most likely relation from the list for the extracted
concepts.
3. output the triplets in the format of (concept 1 ; relation ; concept 2)
strictly.\
\

Provide your answer as follows:
Answer:::
Triples: (The extracted triples)\
Answer End:::\

You MUST provide values for 'Triples:' in your answer.\

"""
```

Without annotated golden references, the evaluation relies on LLM-based judging, a technique for which an LLM is exploited to rate the output in terms of relevance, accuracy and completeness. The evaluation we performed employs Llama-3.1-70B-Instruct[44] as the judge, which ranked in the top 1% on the Huggingface Hub open LLM leaderboard[45] as of January 2025.

The extracted triples are evaluated from the two aspect:

- **Relevance**: Assessment of individual triples' correctness in reflecting contextual information.
- **Coverage**: Evaluation of how comprehensively the extracted triples capture concepts and relationships from the source text.

[44] meta-llama/Llama-3.1-70B-Instruct · Hugging Face

[45] Open LLM Leaderboard - a Hugging Face Space by open-llm-leaderboard

Regarding the relevance, each extracted triple is set to be evaluated individually against the corresponding abstract. For coverage, all extracted triples are evaluated as a set against the corresponding abstract to estimate the completeness of the extracted information. We evaluate the triples on relevance and coverage with two prompts respectively.

Relevance:

```
You are an expert in the field of medical knowledge extraction, focusing on
clinical terminology based on SNOMED CT, an international standard to
represent clinical knowledge.
You are provided with the following information:
- The context, that is, the title and abstract of a scientific paper in the
medical domain
- The extracted knowledge, that is, one triple in the form (subject,
relation, object), automatically extracted from the context
Your task is to evaluate the quality of the extracted knowledge, as
described in the detailed instructions below.

Instructions:
   1. Read the context carefully to identify key medical concepts,
      relationships, and relevant information.
   2. Examine the extracted knowledge by evaluating the given triple.
   3. Evaluate the Relevance of the extracted knowledge on a scale of 1-5,
      focusing on whether it correctly reflects information provided in the
      context. Use the following grading
         - 5: Definitely relevant: the extracted knowledge is clearly
           identifiable in the context.
         - 4: Mostly relevant: the extracted knowledge is identifiable in
           the context, with minor issues.
         - 3: Somewhat relevant: the extracted knowledge is only partly
           identifiable in the context.
         - 2: Mostly not relevant: the extracted knowledge is only
           marginally present in the context.
         - 1: Irrelevant: the extracted knowledge is not identifiable in
           the context.

Please evaluate the following:
context: {abstract}
extracted knowledge: {triple}

Please provide the following output:
Triple Relevance: A numerical score (1-5) assessing the relevance of the
extracted knowledge with respect to the given context.
```

Coverage:

```
You are an expert in the field of medical knowledge extraction, focusing on
clinical terminology based on SNOMED CT, an international standard to
represent clinical knowledge.
You are provided with the following information:
- The context, that is, the title and abstract of a scientific paper in the
medical domain
```

```
- The extracted knowledge, that is, a set of triples in the form (subject,
relation, object), automatically extracted from the context
Your task is to evaluate the quality of the extracted knowledge, as
described in the detailed instructions below.

Instructions:
   1. Read the context carefully to identify key medical concepts,
      relationships, and relevant information.
   2. Examine the extracted knowledge by evaluating the given set of
      extracted triples.
   3. Provide an Overall Coverage Score (1–5). This measures how much the
      extracted knowledge covers the concepts and relationships that can be
      identified in the context. Use the following grading:
         -  5: Complete coverage: the extracted knowledge captures nearly
            all key clinical concepts and relations from the context.
         -  4: High coverage: the extracted knowledge captures many
            clinical concepts and relations from the context, with minor
            omissions.
         -  3: Mild coverage: the extracted knowledge captures part of the
            key clinical concepts and relations from the context.
         -  2: Low coverage: the extracted knowledge misses a significant
            number of key clinical concepts and relations from the context.
         -  1: Null coverage: the extracted knowledge misses almost all key
            clinical concepts and relations from the context.

Please evaluate the following:
context: {abstract}
extracted knowledge: {triples}

Please provide the following output
Triple Coverage: A numerical score (1-5) assessing the coverage of the
extracted knowledge with respect to the given context
```

Results

Table 19: Results of the experiments on Pubmed using Extractor+RAG (Mistral+Bgem3).

Top-K	Extracted Triples	Average Relevance	Average Coverage
30	538	3.438	3.2
50	708	3.416	3.367
100	600	3.728	3.408

Table 19 shows the result of the evaluation on the extracted triples from Pubmed abstracts. The system extracted 538, 708, and 600 triples for Top-K values of 30, 50, and 100 respectively. The peak in extracted triples at Top-K=50 suggests an optimal retrieval window for knowledge extraction. The average relevance scores show a notable pattern, starting at

3.438 for Top-K=30, slightly decreasing to 3.416 for Top-K=50, and then increasing significantly to 3.728 for Top-K=100. This trend indicates that larger retrieval windows may lead to more contextually accurate triple extraction. The coverage metric demonstrates consistent improvement as Top-K increases, rising from 3.2 (Top-K=30) to 3.408 (Top-K=100). This suggests that larger retrieval windows enable more comprehensive knowledge capture from the source abstracts.

The results indicate a trade-off between extraction volume and quality metrics. While Top-K=50 maximizes the number of extracted triples, Top-K=100 achieves the best performance in both relevance and coverage. This suggests that larger retrieval windows may be beneficial for knowledge graph enrichment tasks, particularly when prioritising the quality and comprehensiveness of extracted information.

Case study

The ultimate objective of bottom-up knowledge generation lies in the systematic extraction and integration of knowledge from unstructured textual data into existing knowledge graph frameworks. This process represents a crucial step in expanding structured knowledge repositories through automated information extraction. To demonstrate this knowledge extraction process, we present a specific case study utilizing the research article titled "*Evaluation of Preference and Utility Measures for Transoral Thyroidectomy.*" This example serves to illustrate the practical application of knowledge extraction methodologies in the biomedical domain.

Title: *Evaluation of Preference and Utility Measures for Transoral Thyroidectomy*
Abstract: *Traditional, trans-cervical thyroidectomy results in the presence of a neck scar, which has been shown to correlate with lower quality of life and lower patient satisfaction. Transoral thyroid surgery (TOTS) has been utilized as an alternative approach to avoid a cutaneous incision and scar by accessing the neck and thyroid through the oral cavity. This study was designed to evaluate patient preference through health-state utility scores for TOTS as compared to conventional trans-cervical thyroidectomy.*
DOI: *10.1177/00034894221094*

Table 20: Comparison between the evaluations of the LLM judge and the human expert on the case.

The extracted triples from Pubmed	Relevance score by LLM judge	Human evaluation
(Traditional, trans-cervical thyroidectomy ; has definitional manifestation ; Presence of a neck scar)	5.0	4.0
(Traditional, trans-cervical thyroidectomy ; has realization ; Presence of a neck scar)	5.0	5.0
(Presence of a neck scar ; associated with ; Lower quality of life)	5.0	5.0
(Presence of a neck scar ; associated with ; Lower patient satisfaction)	5.0	5.0

The extracted triples from Pubmed	Relevance score by LLM judge	Human evaluation
(Transoral thyroid surgery (TOTS) ; has definitional manifestation ; Avoid a cutaneous incision and scar)	5.0	4.0
(Transoral thyroid surgery (TOTS) ; is modification of ; Trans-cervical)	2.0	4.0
(Cutaneous incision and scar ; associated with ; Oral cavity)	1.0	1.0
(Health-state utility scores ; measures ; Patient preference)	4.0	5.0
	AVG. 4.0	AVG. 4.125

As Table 20 shows, in this case study of knowledge extraction from the article on transoral thyroidectomy, the evaluation results demonstrate strong alignment between LLM judge and human evaluations. The average scores show comparable assessment quality, with LLM scoring 4.0 and human evaluation averaging 4.125. The evaluation consistency is further evidenced by four out of eight triples receiving identical scores from both LLM and human evaluators, with an average score difference of only 0.625.

The extracted triples effectively capture the core concepts, with several receiving maximum relevance scores (5.0) from both evaluators. These include the relationship between neck scars and patient outcomes, particularly regarding quality of life and satisfaction. The health-state utility measurement triple also received high human evaluation (5.0), validating its importance in the study context.

The triples comprehensively cover the key aspects of the research, progressing logically from traditional thyroidectomy complications to the TOTS alternative. The extraction successfully captures both procedural aspects and patient-centered outcomes, though the relationship between TOTS and trans-cervical procedures received varying scores (LLM: 2.0, Human: 4.0).

Minor variations in scoring appear primarily in technical aspects, such as the definitional manifestations of surgical procedures. These differences suggest that human evaluators may have a more nuanced understanding of medical terminology and procedural relationships, while the LLM maintains a more conservative scoring approach for specialized medical concepts.

3.2.4. Discussion

The comparative analysis of Retrieval-Augmented Generation (RAG) models confirm its value as an information extraction tool, also highlighting a nuanced trade-off between different architectural approaches and configuration strategies.

The experiments on BC5CDR provided a means to compare and select the best RAG configuration. The superior performance of the Extractor+RAG configuration over the All-in-One RAG model suggests that modular approaches can potentially mitigate the limitations of monolithic systems. Our findings indicate that decomposing the information extraction pipeline into distinct extraction and retrieval stages allows for more granular optimization and potentially more robust performance. The Extractor+RAG models consistently outperformed the All-in-One RAG across key metrics, demonstrating improved recall (0.612 vs. 0.559) and marginally better F1 scores (0.694 vs. 0.674). The Top-K setting analysis reveals a significant performance gradient across different retrieval configurations. Moderately high Top-K settings (e.g., 30) demonstrated better overall performance. We hypothesise that the Extractor+RAG approach's superior performance stems from:

1. Enhanced modularity allowing independent optimisation of extraction and retrieval components
2. More flexible handling of information retrieval across varying context complexities
3. Potential reduction of error propagation inherent in monolithic models

Several avenues for future research and model refinement emerge from our findings, including (i) adaptive Top-K strategies that dynamically adjust retrieval depth based on task complexity, (ii) hybrid architectures that combine the strengths of All-in-One and Extractor+RAG approaches, and (iii) computationally efficient methods to maintain high Top-K performance while minimising resource consumption.

The experiments on MIMIC-IV presented a complex landscape for evaluation. Although the quantitative results from the experiments were not particularly promising, a detailed case study revealed significant potential for extracting meaningful insights. This discrepancy can be attributed to the inherent limitations of MIMIC-IV, which was not originally designed for tasks related to information extraction or knowledge graph development. The dataset's structure and focus may have constrained the performance of the models employed. However, the qualitative analysis indicated that there are valuable patterns and relationships within the data that could be harnessed with further refinement of methodologies. This suggests that while immediate results may not meet expectations, there is a pathway forward for enhancing model performance through targeted adjustments in the extraction techniques.

In contrast, The experiments on PubMed yielded more encouraging preliminary quantitative results, as evaluated by both LLM judges and human assessors. The findings indicated a decent level of accuracy in extracting relevant information, reinforcing the reliability of LLM judges as evaluators in this context. Notably, the alignment between LLM judges' assessments and human evaluations underscores the validity of utilizing LLMs for such tasks, providing a promising avenue for future research. The consistency in evaluations suggests that leveraging LLMs could streamline processes in knowledge extraction, making them a viable option for enhancing efficiency in medical literature analysis.

While our analysis provides valuable insights, several limitations should be acknowledged. The study was confined to a specific Mistral LLM configuration, and results may vary across different model architectures and domains. Additionally, the computational resource implications of high Top-K settings warrant further empirical investigation.

The comparative analysis underscores the potential of modular RAG approaches in enhancing information extraction performance. By carefully balancing retrieval comprehensiveness, precision, and computational efficiency, researchers can develop more robust and adaptable information extraction systems.

3.3. Entity Linking

This module is implemented to link the query/term used in the entity extraction experiments. Indeed, the information extraction pipeline produces relations among concepts that are described with human-readable labels. These labels must be linked to the concepts within the knowledge graph in order to be integrated and harmonised with it. Hence, an entity linking step is mandatory. The pipeline for this linking algorithm is depicted in Figure 16. The pipeline takes as input a single label associated with a concept or a relation produced by the information extraction process, and processes the label through a series of software modules to produce a concept ID available in the KG, or an error in case of failure. Each module is described in the following.

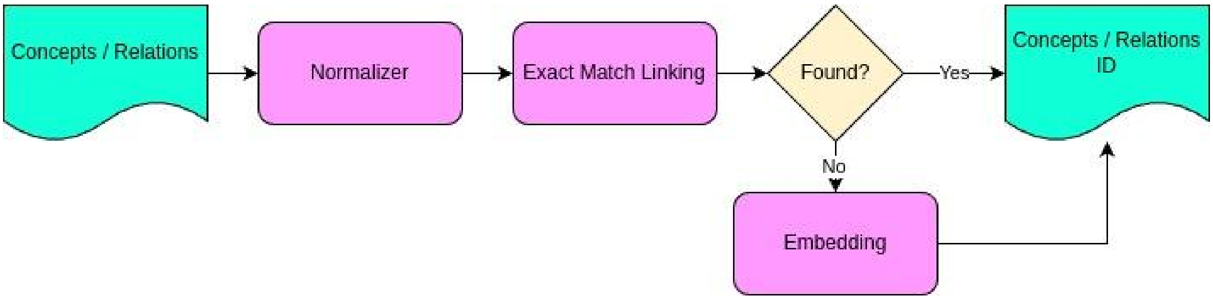


Figure 16. Pipeline of the linking algorithm.

Normalizer. This module takes text input and produces the normalized form of the query input. The normalization procedure relies on the *Norm*[46] pipeline, one of the *Lexical Tools*[47] maintained by the *National Library of Medicine*. Given an input string, *Norm* produces a normalised string taking into account alphabetic case, inflection, spelling variants, punctuation, genitive markers, stop words, diacritics, symbols, ligatures, and word order. Basically, the normalised string is a simplified version of the original text in lowercase, without punctuation, stop words, genitive markers, or diacritics. Words are uninflected, alphabetically sorted, and non-ASCII characters are mapped or converted to ASCII. As demonstration, given the following generated triple from the extraction experiment `(cutaneous incision and scar; associated with; oral cavity)`, this module analyses each label individually and produces the following items:

- `cutaneous incision and scar → cutaneous incision scar`
- `associated with → associated with`
- `oral cavity → cavity oral`

Here we can see that a stopword was removed from the first label, and the alphabetical order was ensured in the third label. With short labels representing extracted concepts, the effects of normalisation are not always apparent, but they anyway lead to a standard form that enables linking afterwards.

[46] https://lhncbc.nlm.nih.gov/LSG/Projects/lvg/current/docs/userDoc/tools/norm.html
[47] https://lhncbc.nlm.nih.gov/LSG/Projects/lvg/current/web/index.html

Exact Match Linking. This module takes the normalized forms of the input text and links them to concepts into the DKG when there is an exact correspondence with any normalized string describing the KG concept. More specifically, the Jaccard similarity between the normalised input label and any normalised label available in the KG is computed, and any pair with similarity 1 is returned. Note that all labels associated to SNOMED concepts in the KG have been normalised and stored as part of the KG itself for fast retrieval.[48] In case of multiple exact matches, all of them are retained (e.g. `Concept1 -> [SctId1, SctId2, ...]`). As a result, the concept ID is associated with the input string:

- `cutaneous incision scar → n/a`
- `associated with → 47429007`
- `cavity oral → 74262004`

Embedding. This module was developed in order to build a robust matching algorithm, complementing the previous module in case the exact match is not available. When this happens, the original string is processed to extract its embeddings and retrieve the closest concepts within the DKG. To this end, the module relies on a vector database where embeddings for all labels denoting SNOMED concepts are stored, based on reference embedding models. As of the time of this writing, we can rely on two state-of-the-art embedding models specifically built for SNOMED. The first, developed by Zahra and Kate [13], covers only SNOMED terms and is based on word-level tokenization and a dictionary of words. However, it does not provide full coverage for all SNOMED concepts and lacks robustness when handling out-of-dictionary input text. The second model, *biosyn-biobert-snomed*[49], is a sentence-transformers model and specifically a fine-tuned version of BERT trained for biomedical and SNOMED-related terminology. It provides wider coverage and subword tokenization enables to process and generate embeddings even for out-of-dictionary terms, making it more suitable for handling a broader range of input text and supporting our entity linking needs.

Semantically, the embeddings can catch the meaning of the terms, hence it is possible to retrieve close concepts. In this case, we exploit the cosine similarity to identify the closest concept to the given input string. For example, `cutaneous incision and scar` on the previous module did not match any concept, but with the embedding module it can be linked to `wound scar(SctId: 286613000)` as follows.

- `cutaneous incision and scar → 286613000`

Inclusion in the KG. Once the linking process is complete, a new set of RDF triples is generated and incorporated into the DKG, along with provenance and linking information. This is possible thanks to the PROV-O ontology[50] for representing provenance and the Evidence core module[51] defined in Deliverable D2.1 for linking information objects (e.g. documents, graphs, or RDF statements) to RDF entities. Figure 17 illustrates the RDF generated for the previous example, that is, the triple `(Cutaneous incision and scar ; associated with ; Oral cavity)`. In this figure, the entity `mdxdata:rag` represents the entire knowledge extraction activity that combines RAG and entity linking, and it is typed as

[48] To this end, entities representing names (i.e., instances of the class `nm:Name` as defined in the Naming ontology module - cf. D2.1) are enriched with a `normForm` property to represent and store the normalised version of the (lexical representation of a) name.

[49] https://huggingface.co/xlreator/biosyn-biobert-snomed

[50] https://www.w3.org/TR/prov-o/ last visited on January 9th 2025.

[51] https://github.com/hacid-project/knowledge-graph/blob/main/ontologies/core/evidence.owl

an instance of the class prov:Activity, as defined by the Provenance ontology. A statement, ex:stmt, records the knowledge generated by the RAG+entity linking pipeline. The statement is an instance of the class ev:Statement, which is defined in the Evidence core module, a part of our ontology network. These statements are associated with (i) a subject (via ev:hasSubject), (ii) an object (via ev:hasObject), and (iii) a predicate (via ev:hasSemantics). Additionally, the statements are linked to mdxdata:rag through the property prov:wasGeneratedBy, thereby recording the provenance of the generated statements with respect to the generation activity.

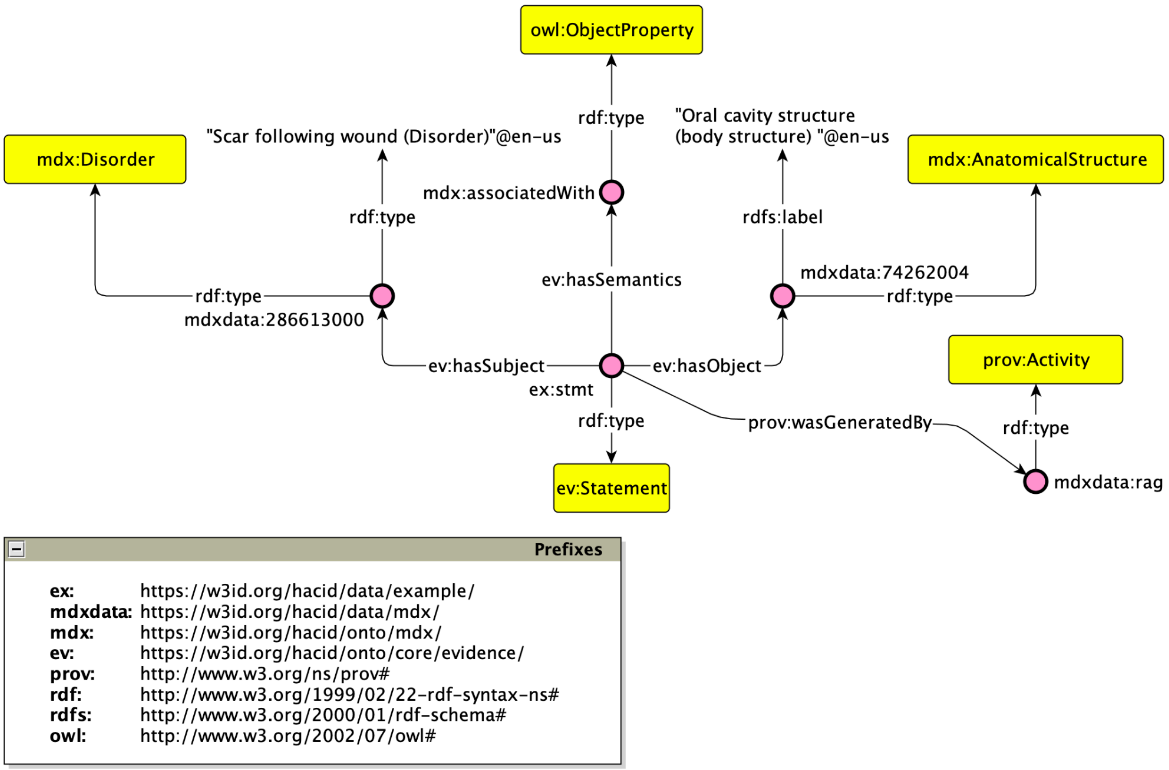

owl:ObjectProperty
rdf:type
"Scar following wound (Disorder)"@en-us
"Oral cavity structure (body structure) "@en-us
mdx:Disorder
mdx:AnatomicalStructure
mdx:associatedWith
rdf:type
rdfs:label
ev:hasSemantics
mdxdata:74262004
rdf:type
rdf:type
mdxdata:286613000
ev:hasSubject
ev:hasObject
ex:stmt
prov:Activity
rdf:type
prov:wasGeneratedBy
rdf:type
ev:Statement
mdxdata:rag
Prefixes
ex: https://w3id.org/hacid/data/example/
mdxdata: https://w3id.org/hacid/data/mdx/
mdx: https://w3id.org/hacid/onto/mdx/
ev: https://w3id.org/hacid/onto/core/evidence/
prov: http://www.w3.org/ns/prov#
rdf: http://www.w3.org/1999/02/22-rdf-syntax-ns#
rdfs: http://www.w3.org/2000/01/rdf-schema#
owl: http://www.w3.org/2002/07/owl#

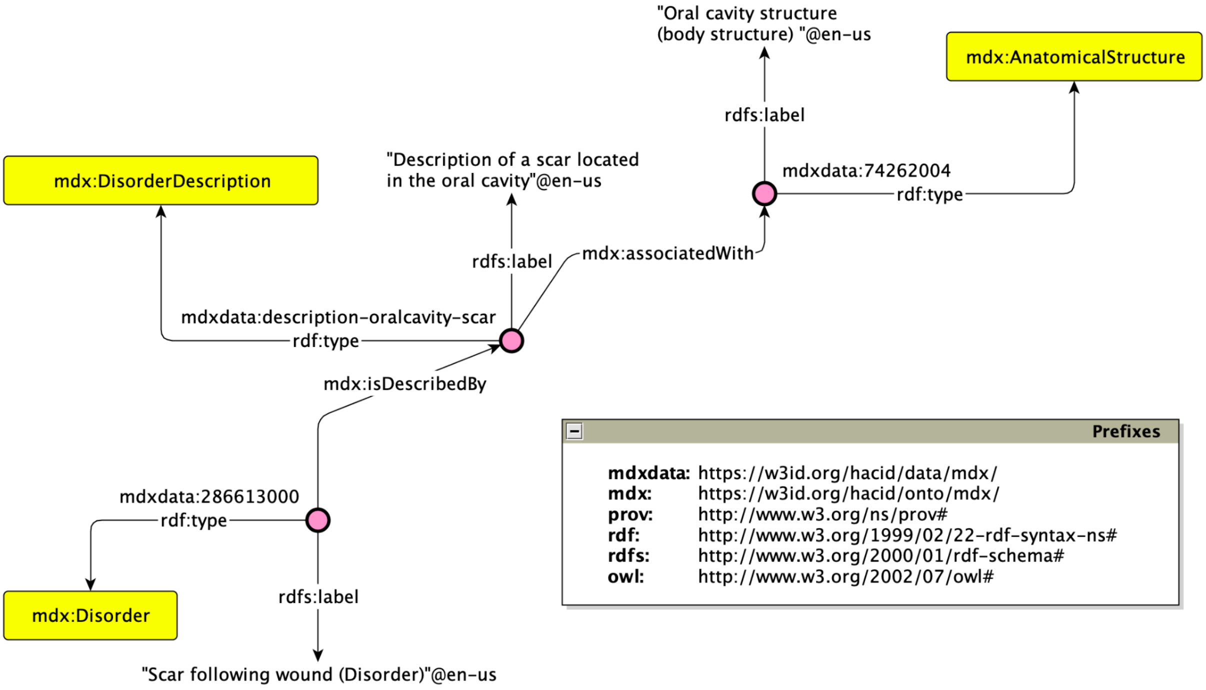
Figure 17. RDF generated with the bottom-up approach with provenance and evidence information.

Reifications, consisting of statements represented as entities linked to each component of a triple, are stored in a dedicated named graph. These reifications can be transformed into more concise triples using a SPARQL CONSTRUCT query, as shown below.

```
PREFIX ev: <https://w3id.org/hacid/onto/core/evidence/>
PREFIX prov: <https://w3id.org/hacid/data/mdx/>
PREFIX mdx: <https://w3id.org/hacid/onto/mdx/>
PREFIX mdxdata: <https://w3id.org/hacid/data/mdx/>

CONSTRUCT {
  ?sbj mdx:isDescribedBy mdxdata:description-oralcavity-scar .
  mdxdata:description-oralcavity-scar a mdx:DisorderDescription ;
          ?pred ?obj
}
WHERE {
  ?activity a prov:Activity .
  ?stmt a ev:Statement ;
        prov:wasGeneratedBy ?activity ;
        ev:hasSubject ?sbj ;
        ev:hasSemantics ?pred ;
        ev:hasObject ?obj .
}
```

The execution of this query generates the RDF shown in Figure 18, effectively harmonising the newly generated knowledge within the knowledge graph. Specifically, the entity `mdxdata:description-oralcavity-wound-scar` describes a clinical finding, `mdxdata:286613000` (wound scar), in terms of its associated finding site (i.e., oral cavity structure). This representation applies the DnS (Description and Situation) pattern[52], as detailed in Deliverable 2.1.



Prefixes

mdxdata:	https://w3id.org/hacid/data/mdx/
mdx:	https://w3id.org/hacid/onto/mdx/
prov:	http://www.w3.org/ns/prov#
rdf:	http://www.w3.org/1999/02/22-rdf-syntax-ns#
rdfs:	http://www.w3.org/2000/01/rdf-schema#
owl:	http://www.w3.org/2002/07/owl#

Figure 18. RDF without provenance and evidence information.

[52] http://ontologydesignpatterns.org/wiki/Ontology:DOLCE+DnS_Ultralite

4. Conclusion

The completion of the medical diagnostics and climate services knowledge graphs represents a pivotal milestone in the HACID project, building upon the robust theoretical and methodological groundwork established in Deliverable 2.1 [1]. By leveraging the top-down and bottom-up knowledge engineering approaches, modular ontology design, and the eXtreme Design (XD) methodology [2], these knowledge graphs exemplify the seamless integration of foundational ontologies and domain-specific requirements.

The medical diagnostics knowledge graph, enriched by data from Wikidata and mappings to SNOMED CT and other established medical ontologies, highlights the potential of ontology-driven integration for enhancing semantic interoperability in healthcare. Similarly, the climate services knowledge graph demonstrates the capability of semantic technologies to model complex climate phenomena and support evidence-based decision-making in environmental policy and adaptation strategies.

The design and instantiation processes documented in this deliverable underscore the scalability, logical consistency, and structural integrity of both knowledge graphs. These qualities not only ensure their immediate utility but also establish a framework for future expansion. In this deliverable we also investigate bottom-up approaches to knowledge engineering. This complementary strategy utilises the abundance of unstructured and semi-structured data in both domains to automatically extract, validate, and integrate knowledge into existing graph structures. For this we designed a solution based on Retrieval-Augmented Generation that provides encouraging results.

Together, these knowledge graphs serve as critical tools for addressing complex challenges in their respective domains. They also stand as an exemplar for applying semantic technologies to foster innovation and improve decision-making, laying the foundation for future knowledge graph initiatives that extend the principles and practices established in HACID.

Potential areas for further development include incorporating emerging data sources, addressing evolving domain needs, and integrating advancements in artificial intelligence for smoothing the integration of top-down and bottom-up approaches to knowledge engineering, such the one envisioned in [14].

References

[1] A. G. Nuzzolese, A. S. Lippolis, W. Xu, M. Ceriani, A. Russo, and V. Trianni, ‘Top-down and bottom-up approaches to domain knowledge engineering’, Dec. 2024, doi: 10.5281/ZENODO.14264690.

[2] 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.

[3] 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.

[4] 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.

[5] V. A. Carriero, A. Gangemi, M. L. Mancinelli, A. G. Nuzzolese, V. Presutti, and C. Veninata, ‘Pattern-based design applied to cultural heritage knowledge graphs: ArCo: The knowledge graph of Italian Cultural Heritage’, *Semantic Web*, vol. 12, no. 2, pp. 313–357, Jan. 2021, doi: 10.3233/SW-200422.

[6] G. Carletti *et al.*, ‘The Water Health Open Knowledge Graph’, 2023, *arXiv*. doi: 10.48550/ARXIV.2305.11051.

[7] A. Gangemi, C. Catenacci, M. Ciaramita, and J. Lehmann, ‘Modelling Ontology Evaluation and Validation’, in *The Semantic Web: Research and Applications*, vol. 4011, Y. Sure and J. Domingue, Eds., in Lecture Notes in Computer Science, vol. 4011. , Berlin, Heidelberg: Springer Berlin Heidelberg, 2006, pp. 140–154. doi: 10.1007/11762256_13.

[8] S. Tartir, I. B. Arpinar, and A. P. Sheth, ‘Ontological Evaluation and Validation’, in *Theory and Applications of Ontology: Computer Applications*, R. Poli, M. Healy, and A. Kameas, Eds., Dordrecht: Springer Netherlands, 2010, pp. 115–130. doi: 10.1007/978-90-481-8847-5_5.

[9] H. Yao, A. M. Orme, and L. Etzkorn, ‘Cohesion Metrics for Ontology Design and Application’, *J. Comput. Sci.*, vol. 1, no. 1, pp. 107–113, Jan. 2005, doi: 10.3844/jcssp.2005.107.113.

[10] A. M. Orme, H. Tao, and L. H. Etzkorn, ‘Coupling metrics for ontology-based system’, *IEEE Softw.*, vol. 23, no. 2, pp. 102–108, Mar. 2006, doi: 10.1109/MS.2006.46.

[11] M. d’Aquin, A. Schlicht, H. Stuckenschmidt, and M. Sabou, ‘Criteria and Evaluation for Ontology Modularization Techniques’, in *Modular Ontologies*, vol. 5445, H. Stuckenschmidt, C. Parent, and S. Spaccapietra, Eds., in Lecture Notes in Computer Science, vol. 5445. , Berlin, Heidelberg: Springer Berlin Heidelberg, 2009, pp. 67–89. doi: 10.1007/978-3-642-01907-4_4.

[12] 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.

[13] F. A. Zahra and R. J. Kate, ‘Obtaining clinical term embeddings from SNOMED CT ontology’, *J. Biomed. Inform.*, vol. 149, p. 104560, Jan. 2024, doi: 10.1016/j.jbi.2023.104560.

[14] A. Gangemi and A. G. Nuzzolese, ‘Logic Augmented Generation’, *J. Web Semant.*, vol. 85, p. 100859, May 2025, doi: 10.1016/j.websem.2024.100859.