HACID - Deliverable

Methods and tools for case knowledge refinement

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:	D3.1
Due date:	29.02.2024
Nature[1]:	R
Dissemination Level[2]:	PU
Work Package:	WP3
Lead Beneficiary:	MPG
Contributing Beneficiaries:	CNR

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

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

Document History

Version	Date	Description	Author	Partner
0.1	15/01/2024	Document creation	Vito Trianni	CNR
0.2	30/01/2024	First draft of methods	Octavianus Sinaga, Nikloas Zoeller	CNR, MPG
0.3	07/02/2024	Refinement of methods	Vito Trianni	CNR
0.4	14/02/2024	Introduction of CKG model	Vito Trianni, Miguel Ceriani	CNR
0.5	23/02/2024	Final methods sections	Nikolas Zoeller, Stefan Herzog	MPG
0.6	26/02/2024	Final NER/NEL methods	Octavianus Sinaga	CNR
0.7	27/02/2024	Final CKG model	Miguel Ceriani, Alessandro Russo, Andrea Nuzzolese	CNR
0.8	29/02/2024	Revision of CKG model	Alessandro Russo	CNR
1.0	29/02/2024	Final revision	Massimiliano Schembri, Vito Trianni	CNR

Table of content

1. Introduction

In this document, we detail the methods employed to define the Case Knowledge Graph (CKG) starting from the information available about a case, either provided by human experts that enrich the case information by means of an interactive dashboard, or automatically extracted by means of state-of-the-art NLP methods.
The starting point of this process is the Domain Knowledge Graph (DKG), which contains the common knowledge about a domain, and can be considered as a relatively static resource (unless updates are routinely performed to ingest new knowledge, e.g., from recent scientific evidence). Whenever the need to provide a solution to a new case arises, a CKG is generated by analysing available information about a case and by relying on expert input. To this end, relevant concepts from the DKG are identified and such information suitably stored so that it can be effectively retrieved for automated reasoning and aggregation of case solutions from expert users.
In the following, we describe the methods envisaged for extraction of concept relevance from automated analysis of case description (Section 2) and from interaction with expert users (Section 3). We then discuss how concept relevance can be expanded following the relations defined in the DKG (Section 4), and how further information can be extracted considering the co-occurence of mentions by human or artificial agents (Section 5). Thanks to these methods, the CKG can be constantly updated thanks to a formal representation that is described in Section 6. Once available, the CKG can be exploited to inform aggregation algorithms for collective case solution, as it highlights the information from the DKG that is relevant for the specific case.

2. Automated Analysis of Case Description

In this section, we describe the methodology that leads to the automatic extraction of structured knowledge from the available case description. Indeed, cases are usually accompanied by rich descriptions of the context in which the case emerged, and this is true for both medical diagnostics and climate services. For instance, a medical case is accompanied by the description of symptoms, the patient's medical records, the results of tests and observations. Following the rapid advancement of natural language processing (NLP) methodologies, it is possible to extract relevant knowledge and map it onto the knowledge graph. Hence, the first approximation of the CKG can be constructed by identifying relevant concepts for the case at hand. To this purpose, the case description is processed to identify relevant entities (named entity recognition, see Section 2.1) and appropriately link them to the KG concepts (entity linking, see Section 2.2), paying attention to the context in which the identified terms appear for proper disambiguation. In Section 2.3 we provide examples of approaches exploited in the biomedical domain, for which valuable models are available in the literature. The climate service domain lacks similar examples, hence within HACID we will transpose methods that work for medical diagnostics to climate services. The chosen methodology is presented in Section 2.4.

2.1. Named entity recognition

Named Entity Recognition (NER) is a widely applicable NLP task that involves identifying and classifying entities (such as name of people, organisations, locations, dates, etc) within a body of text. Once identified, entities are later fed to downstream tasks, such as assertion status detection, entity resolution, relation extraction, and entity linking[3]. For instance, in the medical domain, NER focuses on extracting entities such as genes, protein, diseases, drugs, and more. The pipeline for implementing NER tasks usually starts by preprocessing the input text (e.g., tokenization, noise removal, stemming and lemmatization). Afterwards, features are extracted to provide contextual information for entity recognition. This process usually includes word embeddings, part-of-speech tagging, syntactic dependencies, and more. Following, entity tagging allows to assign a label to each token in the text, indicating whether it belongs to a named entity as well as the type of entity it represents (e.g., person, location, organisation, date, ect).

Background The development of NER has been rapidly increasing over the last decade, especially thanks to the recent advancements in deep learning approaches for NLP. Several methods have been employed to develop NER, from rule-based methods to deep learning approaches and combinations thereof.[4]
Rule-based methods employ heuristics to map text into entities. Dictionary-based methods utilise lexicons and match the exact or approximation of the entities within the text. Regular expression methods define patterns to capture the structure and characteristics of the entities. Machine Learning (ML) has been exploited for NER with different approaches. Conditional Random Fields (CRF) capture the sequential dependencies between words and their labels, making them suitable for NER tasks. Support Vector Machines (SVM) can be trained using features derived from the text to classify each word as a specific entity or non-entity. Hidden Markov Models (HMM) capture the probability distribution over sequences of words and their corresponding labels, making them applicable for sequence tagging tasks. Other approaches make use of deep learning methods. Recurrent Neural Networks (RNN) capture sequential dependencies in the text and are suitable for NER tasks where context is crucial. Long Short-Term Memory (LSTM) Networks, a type of RNN, address the vanishing gradient problem and are effective for capturing long-range dependencies in sequential data. Transformer models like BERT (Bidirectional Encoder Representations from Transformers) have shown significant success in various NLP tasks, including NER.[5]

Advantages and limitations of NER The development of NER has brought major positive impact not only to the NLP field but also to applications in other areas. NER improves the understanding of text content by identifying and classifying named entities, which helps in tasks such as document categorization, sentiment analysis, and information retrieval. It provides context and meaning to the text, enabling better interpretation by machines. NER also plays a crucial role in integrating information from disparate sources by identifying and standardising named entities across different datasets. This facilitates data integration and interoperability in applications such as data mining, knowledge graphs, and semantic web technologies. Furthermore, NER enhances search and retrieval capabilities by enabling users to search for specific named entities or types of entities within textual documents. This improves the accuracy and relevance of search results, leading to better user experience and information discovery.

[3] Kocaman, V., Talby, D. (2020). Biomedical Named Entity Recognition at Scale. 25th International Conference on Pattern Recognition. https://doi.org/10.48550/arXiv.2011.06315
[4] Keraghel, I., Morbieu, S. and Nadif, M., 2024. A survey on recent advances in named entity recognition. *arXiv preprint arXiv:2401.10825.*
[5] Xiaole Li, Tianyu Wang, Yadan Pang, Jin Han, and Jin Shi. Review of research on named entity recognition. In Artificial Intelligence and Security, pages 256–267, 2022.

Although the advancement of NER is remarkable, there are also several challenges which affect the performance of NER tasks. Named entities often exhibit ambiguity and polysemy, where the same text can refer to multiple entities or have different meanings in different contexts. NER systems may struggle to disambiguate such cases accurately, leading to errors in entity recognition. NER systems also may encounter named entities that are not present in their vocabulary or training data, especially in rapidly evolving domains or with domain-specific terminology. Handling out-of-vocabulary entities effectively poses a challenge for NER systems. Furthermore, NER performance may vary depending on the domain or genre of the text. Models trained on one domain may not generalise well to other domains, requiring domain adaptation or fine-tuning for optimal performance. Errors in NER can propagate to downstream tasks and analyses, affecting the overall performance and reliability of NLP applications. Robust error handling and mitigation strategies are necessary to minimise the impact of NER errors on downstream tasks. Ongoing research aims to address these limitations and further improve the accuracy and robustness of NER systems.

2.2. Entity Linking

Named Entity linking (NEL), also known as entity resolution or named entity disambiguation (NED), is the process of connecting entity mentions in unstructured text to their corresponding entities in a knowledge base or reference database[6]. The goal is to disambiguate ambiguous entity mentions by mapping them to unique identifiers representing specific entities. In general there are several steps in the NEL pipeline. First, NER is applied to identify and extract named entities from the text. Once entity mentions are identified, a set of candidate entities is generated for each mention. These candidates are potential matches for the entity mention and are typically retrieved from a knowledge base or reference database that contains entries for known entities. Candidate generation can be based on various factors, including entity type, context, and semantic similarity. The next step is to disambiguate each entity mention by selecting the most appropriate candidate from the candidate set. This is often done by considering various features, such as the context of the mention, the context of surrounding text, the popularity or relevance of candidates, and semantic similarity between the mention and candidates. Machine learning algorithms, statistical models, and rule-based approaches can be used for entity disambiguation. Finally, the selected candidate for each entity mention is linked to its corresponding entry in the knowledge base or reference database. This association provides additional information and context about the entity, such as its attributes, relationships, and metadata.
The methods for NEL can vary depending on the specific task requirements of the application, domain of the data, and available resources such as the availability of annotated training data. Here are some methods usually employed.[7,8]

---

[6] W. Shen, Y. Li, Y. Liu, J. Han, J. Wang and X. Yuan, "Entity Linking Meets Deep Learning: Techniques and Solutions," in IEEE Transactions on Knowledge and Data Engineering, vol. 35, no. 3, pp. 2556-2578, 1 March 2023, doi: 10.1109/TKDE.2021.3117715.
[7] W. Shen, J. Wang and J. Han, "Entity Linking with a Knowledge Base: Issues, Techniques, and Solutions," in IEEE Transactions on Knowledge and Data Engineering, vol. 27, no. 2, pp. 443-460, 1 Feb. 2015, https://doi.org/10.1109/TKDE.2014.2327028
[8] Italo L. Oliveira, Renato Fileto, René Speck, Luís P.F. Garcia, Diego Moussallem, Jens Lehmann, Towards holistic Entity Linking: Survey and directions, Information Systems, Volume 95, 2021, 101624, ISSN 0306-4379, https://doi.org/10.1016/j.is.2020.101624

- Dictionary-based methods: this method involves using predefined dictionaries of entity names and synonyms to match entity mentions in text. This approach is quite straightforward but may lack coverage for entities not included in the dictionary.
- Probabilistic models: statistical models such as Hidden Markov Models (HMMs), Conditional Random Fields (CRFs), or Maximum Entropy Markov Models (MEMMs) can be trained to recognize and link entities based on context and features.
- Machine learning methods: supervised machine learning algorithms, such as Support Vector Machines (SVMs), decision trees, or neural networks, can be trained on annotated data to learn to recognize and link entities.
- Deep learning methods: Neural network architectures such as Convolutional Neural Networks (CNNs), Recurrent Neural Networks (RNNs), or Transformer-based models like BERT and its variants can be trained end-to-end to perform entity linking tasks.

NEL enables more precise and contextually relevant information retrieval by linking the entities mentioned in text with respect to structured knowledge bases. It also helps in disambiguating entity mentions, providing a clearer understanding of the meaning of text and facilitating more semantic analysis. Moreover it supports more advanced and semantically meaningful search functionalities, allowing users to find information based on the context and meaning of entities mentioned in the query.

However, there are several challenges in the development of NEL. Performance heavily depends on the coverage and completeness of the underlying knowledge base. Furthermore, NEL faces domain-specific challenges such as handling the terminology of the entities or abbreviations thereof. Finally, resolving ambiguous mentions or entities with similar names can be challenging, leading to potential errors in linking. It is possible to distinguish between the following cases relevant for disambiguation.

- Polysemy and Homonymy: polysemy refers to the phenomenon where a single term has multiple related meanings whilst homonymy occurs when different terms have the same spelling or pronunciation but different meanings. For example, "SMA" could refer to Spinal Muscular Atrophy or Smooth Muscle Actin.
- Term ambiguity: terms often have multiple meanings depending on context. For example, the term "heart" could refer to the organ, a cardiology-related condition, or metaphorical usage (e.g., "heart of the matter").

Resolving these ambiguities requires considering the specific domain context and relationships between terms.

2.3. Applications to the biomedical domains

Biomedical Named Entity Recognition is the application of NER methods within the field of biomedical research which aims to automatically identify and classify specific biomedical entities or terms with respect to specific goals such as: genes and proteins to understand molecular interactions and biological processes; diseases and medical conditions to aid in studying their prevalence, causes, and potential treatments, drugs and chemical compounds essential for pharmacological and therapeutic research; anatomical terms and organisms to understand biological structures and functions.

Recent developments in biomedical NER vastly involve the utilisation of advanced deep learning techniques, domain-specific pre-training, and large-scale annotated datasets. The most recent state-of-the-art models for biomedical NER are mostly laid under transformer-based architecture such as the BERT model.

The Stanford Biomedical and Clinical Model[9] is a part of the Stanza NLP library, developed by the Stanford NLP Group. Stanza provides a collection of state-of-the-art NLP tools for many languages, including English, Chinese, Arabic, and more. The biomedical model in Stanza provides robust NER capabilities, as it can identify and classify various types of biomedical entities, such as genes, proteins, diseases, drugs, chemicals, species, and more.

BioBERT[10] is a specialised variant of BERT (Bidirectional Encoder Representation from Transformers), specifically designed for biomedical text mining. BioBERT is pre-trained on large-scale biomedical text corpora—including PubMed abstracts and full-text articles—to capture domain-specific knowledge and semantics effectively. This domain-specific pre-training enhances its performance on biomedical text mining tasks compared to general-purpose language models of the same type. **BioNER**[11,12] is a system developed for recognition and normalisation of biomedical entities. The development of this system is built with the BioBERT model for tagging the proper entity classes: diseases, chemicals, genes and protein for each text input.

Biomedical entity linking is a specialised area of NEL that focuses on identifying and linking entities mentioned in biomedical text to entries in biomedical knowledge bases or databases. Biomedical NEL poses unique challenges due to the complexity and variability of biomedical terminology, the presence of domain-specific abbreviations and acronyms, and the need for high accuracy in biomedical applications. Therefore, specialised methods and resources are often employed to address these challenges effectively. For instance, BioBERT can also be seamlessly integrated with external biomedical knowledge bases and ontologies, such as MeSH (Medical Subject Headings), UMLS (Unified Medical Language System), and BioPortal, to facilitate knowledge extraction from biomedical text.

A valuable approach to biomedical NEL is given by **text-SNOMED,**[13] based on the work of Peterson et. al.[14] This method automatically transforms clinical free-text into expressions using SNOMED CT (Systematized Nomenclature of Medicine Clinical Terms). SNOMED CT is a comprehensive clinical terminology system used for capturing and exchanging health information. Beyond entity linking, test-SNOMED also performs relation extraction based on the given text structure and the available relations encoded into SNOMED CT. To this end, the method exploits MetaMap, a NER tool developed by the National Library of Medicine (NLM) to extract Unified Medical Language System (UMLS) concepts from text, as well as BioBERT.

Addressing disambiguation problems in the biomedical domain typically involves applying domain-specific knowledge, ontologies, and natural language processing techniques tailored to the biomedical domain. Machine learning models trained on biomedical text corpora, domain-specific word embeddings, and semantic similarity measures based on biomedical

[9] Zhang Y, Zhang Y, Qi P, Manning CD, Langlotz CP. Biomedical and clinical English model packages for the Stanza Python NLP library. J Am Med Inform Assoc. 2021 Aug 13;28(9):1892-1899. doi: 10.1093/jamia/ocab090. PMID: 34157094; PMCID: PMC8363782.

[10] Jinhyuk Lee, Wonjin Yoon, Sungdong Kim, Donghyeon Kim, Sunkyu Kim, Chan Ho So, Jaewoo Kang, BioBERT: a pre-trained biomedical language representation model for biomedical text mining, *Bioinformatics*, Volume 36, Issue 4, February 2020, Pages 1234–1240, ttps://doi.org/10.1093/bioinformatics/btz682

[11] https://github.com/librairy/bio-ner

[12] Alonso Casero, Álvaro (2021). *Named entity recognition and normalization in biomedical literature: a practical case in SARS-CoV-2 literature*. Thesis (Master thesis), E.T.S. de Ingenieros Informáticos (UPM).

[13] https://github.com/dialoguemd-archives/text-snomed

[14] Peterson KJ, Liu H. Automating the Transformation of Free-Text Clinical Problems into SNOMED CT Expressions. AMIA Jt Summits Transl Sci Proc. 2020 May 30;2020:497-506. PMID: 32477671; PMCID: PMC7233039.

ontologies are commonly employed to tackle these challenges. Additionally, context-aware disambiguation algorithms and rule-based systems may be used to improve the accuracy of disambiguation in biomedical case knowledge. An interesting approach to disambiguation makes use of a cross-domain data integration scheme to augment structural resources and generate a large biomedical dataset for pretraining.[15] A three-part approach for disambiguation is proposed: first, the top 10 candidate entities potentially associated with a mention are extracted by using bi-encoder model that includes a context encoder. Afterwards, a cross-encoder reranking model assigns ranks to each candidate and selects the single most plausible entity. Finally, a post-processing phase allows back-off from the model prediction when the reranking model's score is below threshold. Then, coherent predictions for repeating mentions are computed in a given document by mapping all occurrences of particular mentions to the most frequently predicted entity.

2.4. Methods for Automated Analysis of Case Description

Within the HACID project, given the case description for both medical diagnostics and climate service domains, we implement a pipeline to process the text description to identify the relevant entities and link them to appropriate DKG concepts by taking into consideration the disambiguation of the context for each step (see Figure 1). The specific case description is given as input to the HACID NER/NEL system, together with the DKG on which linking must be applied. The linked concepts then become part of the CKG, as will be elaborated in detail in Section 4. We generally refer to linked concepts as being "mentioned".

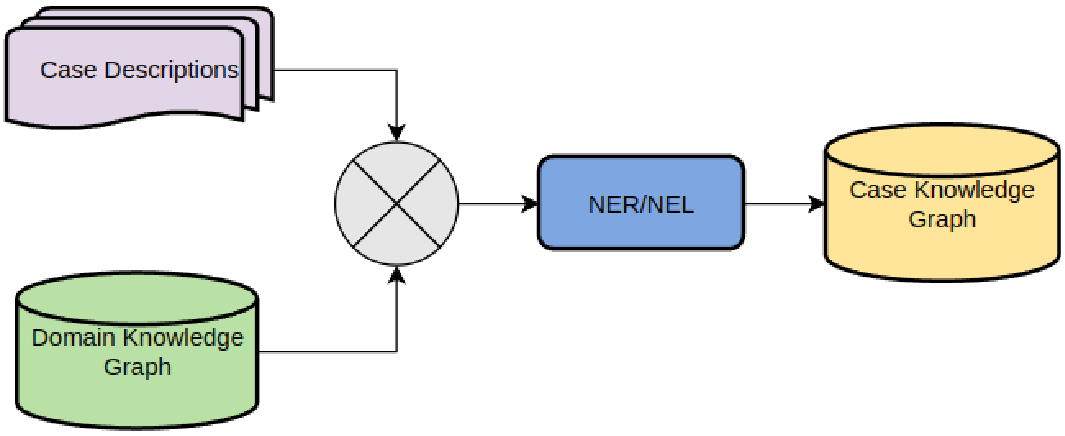


Figure 1. Diagram flow automated analysis process of case description.

For the medical diagnostics domain, the DKG is largely based on SNOMED-CT. Hence, it makes sense to develop NER/NEL approaches that target SNOMED-CT. For the climate services, we will develop a custom DKG and the methods identified here will be translated accordingly. We have selected two approaches for NER/NEL to be tested within HACID. The baseline approach consists in a NER based on the Stanford Biomedical and Clinical Model, followed by a custom string matching algorithm developed for matching text to SNOMED-CT concepts.[16] A more advanced approach is based on text-SNOMED. In this latter approach, the disambiguation issues are already taken into consideration and are therefore minimized.

[15] Varma, Maya, et al. "Cross-domain data integration for named entity disambiguation in biomedical text." *arXiv preprint arXiv:2110.08228* (2021).
[16] Kurvers, R. H. J. M. et al. Automating hybrid collective intelligence in open-ended medical diagnostics. Proc. Natl. Acad. Sci. 120, e2221473120 (2023).

After the initial testing, the methods will be fine-tuned and customised to the final DKG for medical diagnostics, which extends SNOMED-CT in multiple ways. Then, the models we implemented for medical diagnostics are translated to the climate service cases.

3. Gathering case knowledge from users

In this section, we describe the methods devised to gather knowledge from expert users interacting with the DKG/CKG through the data visualisation tool. Such knowledge is intended to refine the domain knowledge with case-specific information that enriches the case description and contributes to the creation of the CKG. Expert users are provided with raw, unprocessed case information and are exposed to the DKG, or to the CKG resulting from automated analyses of the case description (see Section 2). The visualisation tool enables to explore the domain knowledge and identify concepts that are relevant for the case—hereafter referred to as mentioned concepts—as well as to locate missing information that the user deems worth including. The mechanisms that enable users to explore the DKG through the visualisation tool are out of the scope of the present document. Here, we focus on the methodologies that support enrichment or refinement of the case knowledge, to be mirrored in the CKG. This will be possible through (i) concept tagging, (ii) free-text commenting, and (iii) ability to suggest new content. All these methods enable users to organise information within a particular case in a way that makes sense to them, hence supporting cognitive offloading of users' ideas into the CKG.
Concept tagging enables users to mark concepts that are relevant for the given case. Target concepts are treated in a similar way as with the concepts resulting from automatic analyses of case descriptions, the only difference consisting in the fact that these concepts are associated with a human expert.
By means of concept tagging, users can identify relevant information for the given case, but constrained to the existing information encoded in the DKG. However, users may also recognise the need to add new information to the knowledge graph. This can be done in multiple ways. Proficient users could directly introduce structured knowledge within the CKG. To this end, the visualisation tool could offer masks to select the possible classes and relations that are allowed by the DKG. The same knowledge could be suggested for later inclusion in the DKG, shall it be deemed relevant by the user. In alternative, users can exploit the possibility of producing free-text comments, which can be linked to existing concepts or generally associated to the case as a whole. Free text comments are analysed with the same information extraction pipeline discussed in Section 2, leading to the identification of concepts that are relevant for the case, as suggested by the user. Novel structured information could be discovered in this case. For instance, new concepts or relations that are not yet available in the DKG could be extracted from the text and made available in the CKG, possibly upon validation by the user. To this end, we exploit the same information extraction approach developed for the bottom-up construction of the DKG, as detailed in deliverable D2.1.
Finally, users may want to suggest relevant evidence from scientific or grey literature that contains information relevant to the case. Also in this case, we exploit the methodology for bottom-up construction of knowledge graphs devised for generation of structured knowledge. Such information is also later reflected in the DKG, after appropriate scrutiny.

4. Diffusion methods for concept relevance

The problem we want to address in this section is the following: after a relevant concept has been mentioned (automatically or by an expert), how can the DKG be leveraged to identify other concepts that were not explicitly mentioned but might be relevant and how can their relevance be estimated? One way to tackle the above problem is to define a distance metric $d(c_i, c_j)$ on the the DKG through which then also the similarity $s(c_i, c_j) \in [0, 1]$ between two concepts *i* and *j* on the graph is quantified. Additionally, some sort of cutoff threshold t is introduced which confines the considered nodes to some neighbourhood $N_i = \{c_j : d(c_i, c_j) \leq t\}$ so that only the concepts in the neighbourhood N_i and not all concepts on the DKG (which often might be infeasible) are assigned a relevance score. This is schematically shown in Figure 2. If a concept c_i is mentioned (e.g., by an expert), it is assigned an initial relevance score w_i. Then, other concepts are assigned a relevance score according to their similarity to the initial concept c_i:

$$\forall\, c_j \in N_i,\ w_j = w_i\, s(c_i, c_j)$$

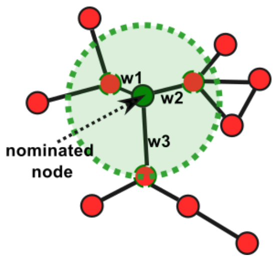


Figure 2: Schematic representation of relevance diffusion on the knowledge graph. Note that the green circle, representing the threshold of the diffusion, only exists if the graph is embedded in a vector space. Otherwise the threshold distance is only defined on the graph itself (for example by the number of hops).

A straightforward simple example of a distance metric is the number of hops h to reach one concept node from the other via the shortest path on the (undirected) DKG, i.e. $d(c_i, c_j) = h_{ij}$ and a possible corresponding similarity metric could be defined as $s(c_i, c_j) = 1/(1 + h_{ij})$ where the threshold distance might be set to $t = 2$ hops. A more complicated approach could involve creating node embeddings, for example via node2vec[17], to map nodes onto vectors in a vector space for which then a variety of distance metrics like the euclidean distance exist. In a similar direction, knowledge graph embeddings could be leveraged to exploit the semantic structure of the knowledge graph, translating concepts in a semantic-rich vector space.[18] On such vector spaces, it is then straightforward to construct a similarity metric, for example $s(v_i, v_j) = 1 - d(v_i, v_j)/t$ if $d(v_i, v_j) \leq t$ else 0. A common

[17] Grover, Aditya, and Jure Leskovec. "node2vec: Scalable feature learning for networks." *Proceedings of the 22nd ACM SIGKDD international conference on Knowledge discovery and data mining*. 2016.

[18] Jiahang Cao, Jinyuan Fang, Zaiqiao Meng, Shangsong Liang, Knowledge Graph Embedding: A Survey from the Perspective of Representation Spaces. arXiv:2211.03536, 2023.

alternative is to use cosine-similarity which is just based on the angle and not on the full euclidean distance in that vector space.
We regard the choice of distance metric as well as the cutoff parameter as hyper parameters that can be tuned to suit the specific use case and purpose of the application.

5. User-concept graph representation

While the last section has described how to leverage the structure of the DKG to identify relevant concepts starting from a seed concept, this section expands this approach and also includes the users into a bipartite user-concept graph.

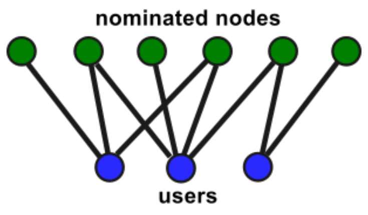


Figure 3. A bipartite graph between users and mentioned nodes of the DKG.

5.1. User-concept bipartite graph

In a bipartite graph, the set of all nodes can be divided into two disjoint and independent sets U and V. Nodes in either of those sets do not share any edges, instead edges only connect nodes from U with nodes in V. A bipartite graph is a special case of a multilayer graph in which different types of nodes can be connected to all other nodes (both of the same and other types) and by different types of relations. As an example consider users interacting with the DKG/CKG through the data visualisation tool, tagging relevant concepts as described in Section 3. Taken together, users and concepts span a bipartite graph (see Figure 3), similar to the user-concept bipartite graph described in Herzog & Hills (2019)[19].
In Section 2, we described how concepts can be automatically linked by analysing the case description. By regarding this step as an artificial agent mentioning relevant concepts, we can integrate this agent and its tagged concepts into the bipartite graph. Recently, it has been reported that commercial large language models (LLMs) like GPT4 have surpassed human performance across diverse tasks. Such an LLM constitutes another type of artificial agent that can be integrated in the user-concept graph by simply passing the case description to the LLM and prompting it to identify additional relevant concepts which can subsequently be mapped back to the DKG/CKG.

5.2. Bipartite graph expansion using diffusion methods of concept relevance

As outlined in Section 4, we can expand the set of relevant concepts via the structure of the DKG. This methodology can also be applied to the bipartite user-concept graph in the following way. For each concept node in the concept layer of the bipartite graph, we identify

[19] Herzog, S. M., & Hills, T. T. (2019). Mediation centrality in adversarial policy networks. Complexity, 2019, 1–15. https://doi.org/10.1155/2019/1918504

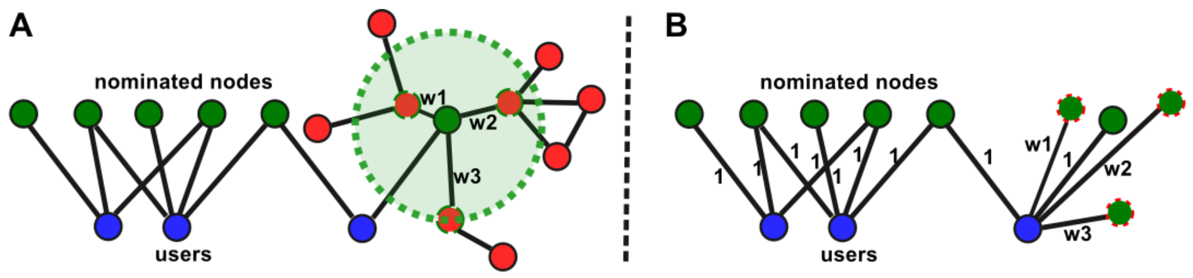


Figure 4. Methods for bipartite graph expansion. A) shows the constructed bipartite graph and the relevance diffusion from one of the mentioned seed nodes on the DKG as described in section 5.1. B) The bipartite graph is expanded by creating edges between the diffused nodes and the user who mentioned the seed node. (Note that for visual clarity only the expansion for the right-most mentioned node via the DKG is illustrated and how this affects the bipartite graph)

relevant nodes and their corresponding weights through relevance diffusion as described in section 4 (see Figure 4A). Each newly identified relevant node is then connected by a weighted edge to the user who originally mentioned it in the annotation process (see Figure 4B). The users in the bipartite graph can be either human experts or artificial agents as described in Section 3. In principle they can be treated the same and assembled in a single bipartite graph, but depending on the specific domain and use case of the resulting decision support system it can also make sense to build separate bipartite graphs (and subsequent projections) for both types of agents or weight them differently.

5.3. Concept graph: Projection of bipartite graph onto concepts

By projecting the user layer onto the concept layer in the bipartite graph, we use information from the relations between users and concepts to create new edges between different concepts. Here we follow a methodology developed by Herzog & Hills (2019)[20].
The simplest approach is to create an edge between two concept nodes if they were both mentioned by the same user, as illustrated in Figure 5. For the adjacency matrix of the new graph this means a cell is set to 1 if at least one user has mentioned both nodes. This projection can also be done in a weighted manner by increasing the weight of an edge between two concepts to *n* if both concepts were mentioned by several users and by accounting for the weight of an edge (e.g., expanded nodes might have lower weights than nodes actually mentioned by users as illustrated in Figure 3B).
In the resulting projected concept graph, the importance of a node can be determined by using a network node centrality measure suitable for the application at hand, e.g., degree centrality, eigenvector centrality, PageRank or in-betweenness centrality; all of which can also be calculated with weighted edges. We call this measure **co-nomination centrality**. It can be integrated in the CKG as an additional property quantifying the importance of a concept. Note, that this is a different measure compared to the relevance ranking mentioned in section 5 which is based on the concept tagging by an expert (and taking the

[20] Herzog, S. M., & Hills, T. T. (2019). Mediation centrality in adversarial policy networks. Complexity, 2019, 1–15. https://doi.org/10.1155/2019/1918504

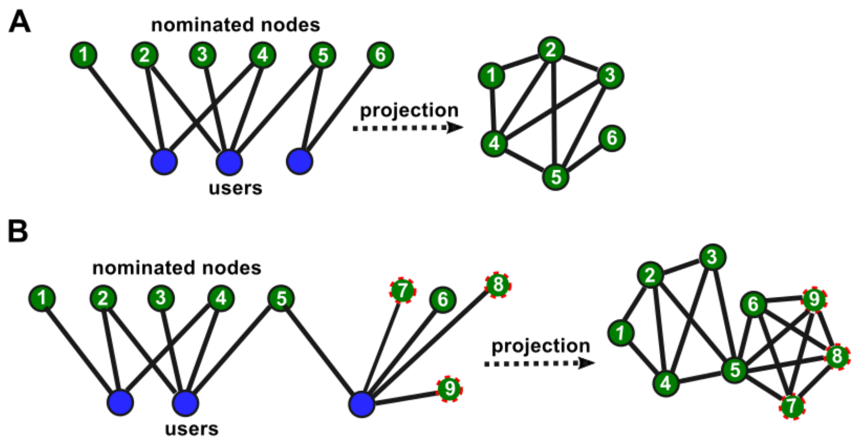


Figure 5. A) Projection of the bipartite graph into a simple graph among concepts. B) Projection of the bipartite graph after an expansion has been performed by creating edges between the diffused nodes and the user who mentioned the seed node (see Figure 4B).

corresponding relevance diffusion into account). A final relevance score might be a combination of both depending on the specific use case.
One problem that might arise with this approach is that, if a user mentions many concepts, these will become more central in the projected graph, because they receive many edges. Depending on the final purpose of the importance ranking this might not be sensible. To address this, we will explore normalising weights on the user level so that each user either brings the same amount of total edge weights into the projected graph (i.e., unity normalisation) or that total weight a user brings into the graph is modelled as a function of additional information (e.g., the number of mentioned nodes as a proxy for expertise).
Finally, we will explore different and more complex projection mechanisms to construct the adjacency matrix of the new concept graph. In particular, this will include creating edges between mentioned concepts conditional on rules that apply to the knowledge graph. An example of such a rule might be that they share a common parent node on the DKG or that both have a certain property.

6. Representation of Case Knowledge

A case knowledge graph initially includes descriptive knowledge about a case. The initial case description, and thus the corresponding case knowledge graph, then evolves and gets refined as new information is collected or becomes available, also as a result of the interplay between expert users and the HACID DSS.

6.1. Case Knowledge Graph creation

Given a particular case, the initial case description is typically provided by the experts that are involved in the case (as in our climate service scenario), or by considering structured information produced by a system where the case was first described (the HumanDX application in our medical diagnostics scenario).
Although a case can be described and represented with general properties and attributes (a name/title, a textual description, a creator, a date, etc.) regardless of the specific domain, detailed case descriptions are clearly domain specific. To support the representation of

domain-specific case knowledge, and thus create Case Knowledge Graphs, Deliverable D2.1 defines domain-specific ontology modules for representing a case in the medical diagnostics and climate services scenarios.
Clinical cases can thus be described with information about the patient and its symptoms, or, more generally, case-related findings. Similarly, a case in the climate service domain is described by details about the organisation requesting the service and the description of the extant request. Case descriptions may already come with references to entities in the corresponding domain knowledge graph. In the medical diagnostics scenario, differential diagnoses can directly refer to SNOMED concepts representing diseases. In the climate services setting, the user describing a case through a dedicated web form can select and reference domain concepts (e.g., contextualising the case with a reference emission scenario, or refer to specific indicators from a controlled vocabulary) as part of the case description process. As a consequence, a case knowledge graph will already have links to, and be integrated with, the domain knowledge graph when it is first created.

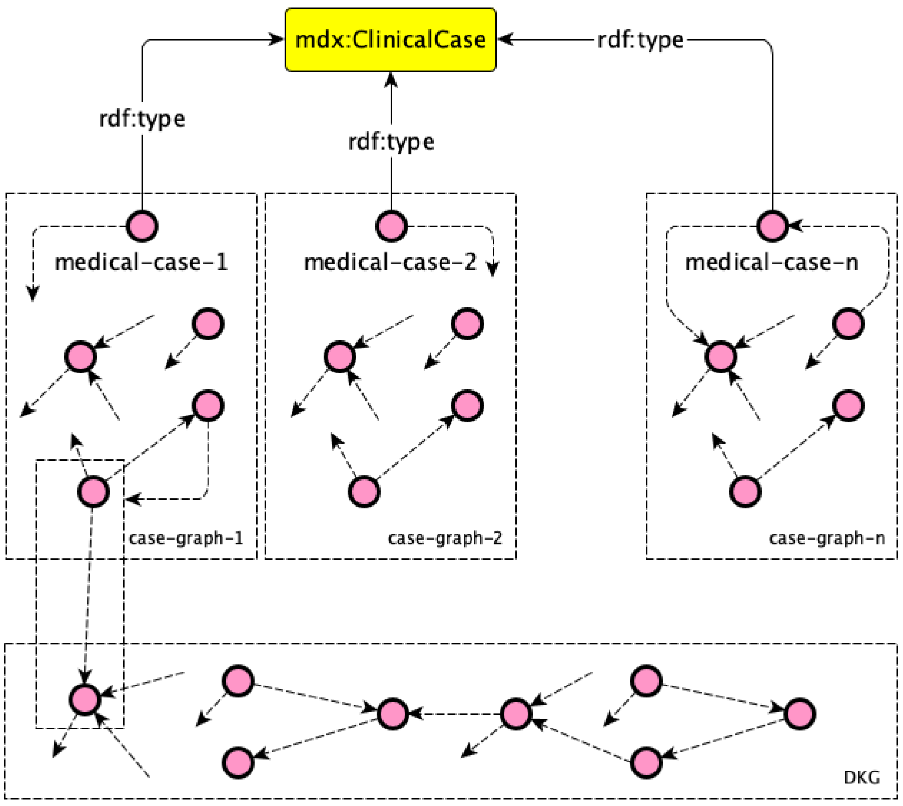


Figure 6. Named graphs as a modular approach to partition case knowledge

Each case, when represented as a case knowledge graph, will have as reference entity and entry point to related knowledge an entity representing the case itself, as an instance of the mdx:ClinicalCase or cs:ClimateCase classes defined as part of the domain ontologies for the two use case scenarios (as defined in D2.1). When representing case knowledge graphs as RDF graphs, it is convenient to rely on RDF named graphs[21] to group together all knowledge (as represented by RDF triples) pertaining to a specific case in at least a dedicated named graph. Each named graph serves as a container for related case knowledge (or a subset

[21] Carroll J.J., Bizer C., Hayes P., Stickler P. (2005). Named graphs. J Web Semant, 3 (4), pp. 247-267, 10.1016/j.websem.2005.09.001

thereof, as the case of the agent-concept graphs described later in this section), providing a clear context for case-specific triples. The fact that named graphs are identified by URIs also enables referring to a case knowledge graph as an entity, so as to make additional assertions that qualify the entire graph. This facilitates representing metadata about a case (e.g., when the information about a case was last updated), provenance information (to represent, for example, that groups of triples were derived from a given source, or produced by a specific agent), and other semantic annotations. Named graphs can also include metadata related to the uncertainty or confidence levels or relevance associated with the information contained within them, by referring to named graphs in statements concerning relevance (as discussed below). Figure 6 provides an informal graphical representation of these concepts.

6.2. Case Knowledge Graph refinement

As discussed in the previous sections, various approaches and techniques can be employed to establish relationships between case-specific knowledge and domain knowledge. Whether this relies on automated NLP pipelines (as presented in Section 2) or is the result of a user-initiated manual process (as outlined in Section 3), the overall objective is to create and represent rich semantic links between a case and concepts in the domain knowledge graph.

6.2.1. Linking case knowledge and domain knowledge

Links between case knowledge and domain knowledge often have to include additional information regarding the identified concepts from the DKG with respect to the case being analysed. What is conceptually a (potentially weighted) link between a case and a concept within the DKG, is more accurately defined as a complex situation involving multiple entities with various roles. Specifically, from a knowledge representation perspective, we need an ontological model that allows representing the following core elements:

- the **entity** to which a concept from the DKG is being related, i.e., a **case**; specific entities that are part of a case can be considered as well (e.g., the case description), to relate concepts to specific elements of a case;
- the **concept** that is being related to the case; any possible entity that exists in the DKG can be considered in relation to a case (e.g., a disease, a climate model, or even a previously addressed case that could be deemed relevant for the case at hand, represented as a specific entity or even as a complex named graph) as a result of an assessment process;
- the **agent** that establishes the relation between the case and the concept, also defining the relevance where applicable; this can be a software agent, such as a NER/NEL toolkit, or a user that manually relates a concept to a case;
- the **relevance** that qualifies the relationship between the concept and the case.

Representing situations where an agent evaluates or judges the relevance of an entity in a specific setting is a recurring requirement in our reference scenarios. Therefore, in Deliverable D2.1 we have defined a general modelling solution or design pattern to address this need, as part of the core modules of the ontology network. Specifically, the *Judgement* ontology module[22] allows representing a judgement situation (`jdg:Judgement`) that involves at least the *entity* object of the judgement, the *agent* that expresses the judgement,

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

and the *result* of the judgement, which can also be used to express relevance in case the judgement involves the assignment of relevance (`jdg:RelevanceAssignment`). Judgements can then be combined by relying on a `jdg:MetaJudgement`, where the entity object of the judgement is itself a judgement.

Representing the target situation outlined above requires, according to our reference ontology module, to rely on a `jdg:RelevanceAssignmentOnJudgement`, which allows defining the relevance (as a `RelevanceAssignment` judgement) on a judgement (i.e., a `MetaJudgement`) that in turn relates an entity (a case, in our setting) with a judgement result represented by the concept we want to relate to the reference entity/case. In essence, to represent the relevance of a concept with respect to a case as defined by an agent, we resort to a "layered" knowledge representation approach where:

1. a first judgement situation allows representing the fact that a specific concept is considered relevant for the given case by an agent;
2. a second judgement situation (specifically, a `RelevanceAssignmentOnJudgement` situation) allows assigning a scored relevance to the previously defined judgement situation, according to the same agent.

The example shown in Figure 7 represents the scenario where the user Claire considers the disorder “Viral pneumonia” (as defined in the medical DKG) as having a relevance of 0.7 with respect to a given case.

Notice that if there is no need (or no information is available) to quantitatively or qualitatively define the relevance, a single judgement situation is needed to assert the fact that a concept is relevant for a case. The modularity of the approach then allows introducing additional knowledge (with the definition of a second situation that assigns relevance to the first one) if and when the need to qualify or quantify the relevance emerges. Such an approach also accommodates the need to take into account a certain degree of subjectivity, as different individuals or algorithms may prioritise different concepts based on their interpretation of the case. Similarly, if the computed or assigned relevance needs to be further qualified to specify the type of relevance that is considered—for instance to specifically represent the co-nomination centrality of a concept—a `Relevance` can be assigned a type (basically following a classification design pattern[23]) or the class can be specialised to represent specific types of relevance.

Situations in which users (or in general agents) relate concepts to specific entities resemble the act of *tagging*, as also mentioned in previous sections. Both defining the relevance of a concept for a case and the act of tagging aim to associate a piece of information (the concept or the tag) with a specific context (the case or the item being tagged). Tags and relevant concepts establish semantic connections between items and knowledge elements, in an attempt to convey the semantic relationship between domain knowledge and the specifics of a case, potentially providing insights into how domain expertise can be applied to address the case effectively. Various ontologies[24,25,26] and Ontology Design Patterns (ODP)

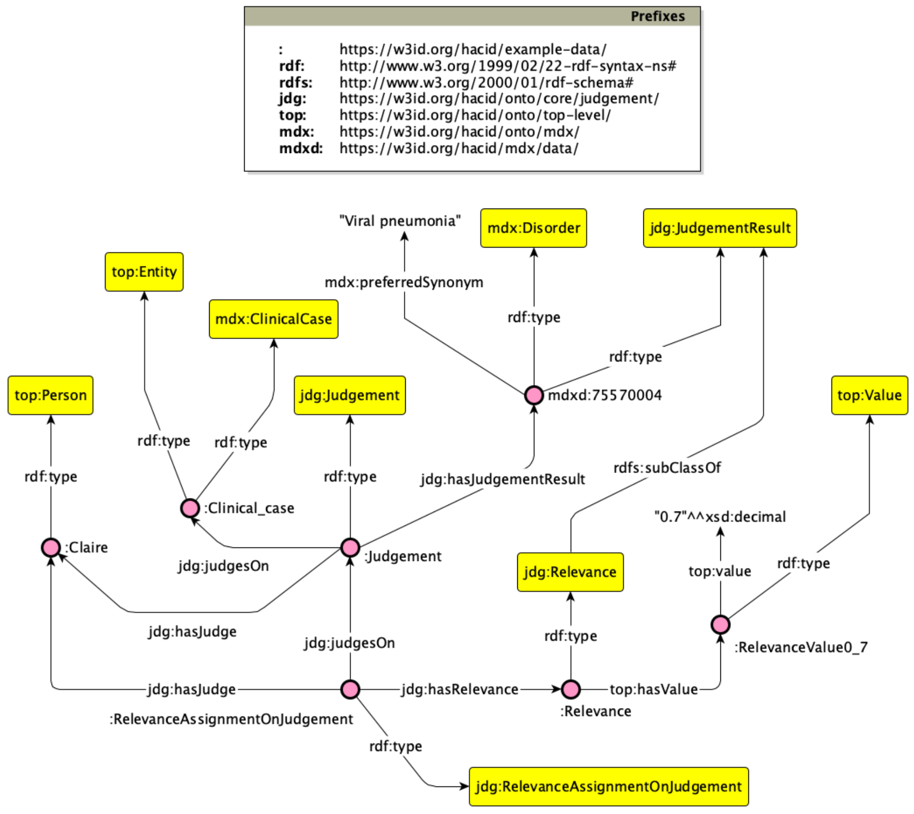
[23] http://ontologydesignpatterns.org/wiki/Submissions:Classification

[24] T. Knerr (2006). “Tagging ontology-towards a common ontology for folksonomies.” https://storage.googleapis.com/google-code-archive-downloads/v2/code.google.com/tagont/TagOntPaper.pdf

[25] F. Limpens et al. (2009). “NiceTag Ontology: tags as named graphs”. In: Proceeding of ASWC 2009 - 4th Asian Semantic Web Conference. https://hal.science/hal-00434129/document

[26] S. Lohmann et al. (2011). “MUTO: the modular unified tagging ontology”. In: Proceedings of the 7th International Conference on Semantic Systems I-SEMANTICS. pp. 95–104. http://doi.acm.org/10.1145/2063518.2063531

have been proposed to represent a tagging action, i.e., the action performed by an agent that attaches a label or an entity with a well-defined semantics (e.g., a concept from a reference controlled vocabulary or domain ontology) to some entity. In the Tagging Ontology Design Pattern,[27] a tagging situation relates together a tag, a tagged thing, a source folksonomy from which the tag is taken, a tagger agent, a polarity, and a temporal reference at which the tagging happens. The overview on tag ontologies provided by Hak Lae Kim et al.[28] is still relevant to understand the proposed approaches. In most of them, in line with the Tagging ODP mentioned before, the tagging is represented as a reified *n*-ary relationship between the tagger/agent, the tag, the entity tagged, and the date when the action happened.



Prefixes

:	https://w3id.org/hacid/example-data/
rdf:	http://www.w3.org/1999/02/22-rdf-syntax-ns#
rdfs:	http://www.w3.org/2000/01/rdf-schema#
jdg:	https://w3id.org/hacid/onto/core/judgement/
top:	https://w3id.org/hacid/onto/top-level/
mdx:	https://w3id.org/hacid/onto/mdx/
mdxd:	https://w3id.org/hacid/mdx/data/

Figure 7. KG representing the scenario where a user (`:Claire`) considers the "Viral pneumonia" disorder (`mdxd:75570004`) as having a relevance of 0.7 with respect to a given case (`:Clinical_case`).

Approaches to represent tagging situations and actions can be considered as specialisations of our general approach for representing relevance as judgements. Although the temporal perspective (often present in tagging ontologies) is not explicitly defined as a core element of our modelling solution, this does not prevent enriching judgement situations and relevance assignments with temporal information (thus considering a `Judgement` as `TimeIndexedSituation`[29]). The inclusion of a temporal dimension allows dealing with scenarios where relevance can evolve over time or vary depending on the changing

[27] A. Gangemi. https://w3id.org/lode/http://ontologydesignpatterns.org/cp/owl/tagging.owl
[28] Hak Lae Kim et al. (2008). "The state of the art in tag ontologies: a semantic model for tagging and folksonomies". In: Proceedings of the International Conference on Dublin Core and Metadata Applications. pp. 128–137. https://dcpapers.dublincore.org/files/articles/952109240/dcmi-952109240.pdf
[29] http://ontologydesignpatterns.org/wiki/Submissions:TimeIndexedSituation

circumstances of the case, and it could be useful to keep track of the various judgements and relevance assessments of concepts as new information becomes available or the case progresses.

6.2.2. Creating and representing user-concept graphs and projected concept graphs

Our knowledge representation approach to relate concepts to cases taking into account the relevance defined or perceived by the involved agents represents the starting point to support the definition of bipartite agent-concept graphs, as explained in Section 5. Bipartite agent-concept graphs are in fact naturally induced by the `Judgement` situations: basically, every instance of a `Judgement` that involves an agent acting as the judge and a concept representing the result of the judgement corresponds to a logical link between the agent and the concept. These links, corresponding to chains or combinations of property paths in the knowledge graph, can be identified and made explicit by defining a SPARQL query that, for a given case, identifies all pairs of agents and concepts (i.e., the edges in the bipartite graph) related by a `Judgement`. The reference structure of such a SPARQL, that basically performs a projection of agents and concepts, is provided hereafter.

```
PREFIX jdg: <https://w3id.org/hacid/onto/core/judgement/>

SELECT DISTINCT ?agent ?mentionedConcept
WHERE {
  ?judgement a jdg:Judgement ;
             jdg:isJudgementOn :case-1234 ;
             jdg:hasJudge ?agent ;
             jdg:hasJudgementResult ?mentionedConcept .
}
```

While the query above generally refers to agents and does not impose constraints of the type of agent that performs a judgement, it can be easily refined and specialised to build separate bipartite graphs for human experts and artificial/software agents, as reported in Section 5.2. The following SPARQL query considers only judgements made by humans, thus focusing on a user-concept bipartite graph.

```
PREFIX : <https://w3id.org/hacid/example-data/>
PREFIX jdg: <https://w3id.org/hacid/onto/core/judgement/>
PREFIX top: <https://w3id.org/hacid/onto/top-level/>

SELECT DISTINCT ?person ?mentionedConcept
WHERE {
  ?judgement a jdg:Judgement ;
             jdg:isJudgementOn :case-1234 ;
             jdg:hasJudge ?person ;
             jdg:hasJudgementResult ?mentionedConcept .
  ?person a top:Person .
}
```

The query can then be further specialised and refined as needed, defining additional constraints on the involved entities and judgements to be considered for the bipartite graph (e.g., restricting the selection to users with certain expertises or characteristics, or considering concepts of a specific type only, or defining thresholds on the relevance, or even combining multiple criteria).

While a `SELECT` SPARQL query produces a bipartite agent-concept graph in the form of a result set made of agent-concept pairs, a bipartite agent-concept graph can also be made explicit and created as an RDF graph, relying on a `CONSTRUCT` SPARQL query that creates direct links (using for example a `jdg:mentions` property that can be added as part of the Judgement core ontology module) between agents and concepts. The following SPARQL query creates for a given case a bipartite agent-concept RDF graph that includes triples of the form `top:Agent jdg:mentions top:Concept`.

```
PREFIX : <https://w3id.org/hacid/example-data/>
PREFIX jdg: <https://w3id.org/hacid/onto/core/judgement/>

CONSTRUCT {
      ?agent jdg:mentions ?concept
}
WHERE {
  ?judgement a jdg:Judgement ;
            jdg:isJudgementOn :case-1234 ;
            jdg:hasJudge ?agent ;
            jdg:hasJudgementResult ?concept .
}
```

Result sets or RDF graphs produced by `SELECT` and `CONSTRUCT` SPARQL queries can then be further processed as outlined in Section 5. Bipartite agent-concept graphs can be also materialised and stored back in the reference triplestore, relying on the SPARQL 1.1 Update Language, so that the triples representing a bipartite agent-concept graph are inserted into a dedicated named graph as a result of an `INSERT` operation, as sketched in the following.

```
PREFIX : <https://w3id.org/hacid/example-data/>
PREFIX jdg: <https://w3id.org/hacid/onto/core/judgement/>

INSERT {
  GRAPH <https://w3id.org/hacid/example-graph/case-1234-agent-concept> {
      ?agent jdg:mentions ?concept
  }
}
WHERE {
  ?judgement a jdg:Judgement ;
            jdg:isJudgementOn :case-1234 ;
            jdg:hasJudge ?agent ;
            jdg:hasJudgementResult ?concept .
}
```

The creation of a dedicated named graph to "contain" all triples defining the initial bipartite agent-concept graph for a case is then the starting point for the graph expansion and projection techniques defined in the Section 5, that produce additional knowledge/ edges/triples to be added to the named graph.

It is worth highlighting that representing bipartite graphs (as well as expansions and projections thereof) by introducing direct links between agents and concepts is appropriate only as long as there is no need to further characterise such links/edges. As soon as the links connecting agents and concepts (or concepts with other concepts, as a result of a projection) need to be represented as weighted edges, the links between entities have to be represented as a class rather than a property[30], so that individual instances of such a class correspond to instances of the relation and can be further qualified by defining a score/weight or other attributes that describe a relation. Such an approach is already adopted in the core ontology module for representing judgements and relevance assignments. The combination of judgments and relevant assignments, as provided by `RelevanceAssignmentOnJudgement` situations, are again a viable option for representing weighted relations among entities. In line with the representation solution outlined above for defining the relevance of concepts for a case, representing weighted edges between agents and concepts resulting from bipartite graph expansion steps requires to:

- define a `Judgement` situation for representing the fact that a specific concept (i.e., a newly identified relevant or diffused node) is considered relevant for a case by an agent (the agent who mentioned a seed node from which diffused nodes are considered);
- define a `RelevanceAssignmentOnJudgement` situation to assign a relevance or weight to the previously defined judgement situation, according to the same agent.

Similarly, a weighted edge between two concept nodes can be represented combining:

- a `Judgement` situation for representing the fact that a concept is considered relevant in relation to another concept by an agent (the software agent that is performing the projection and computing the weights);
- a `RelevanceAssignmentOnJudgement` situation to assign a scored relevance or weight to the previously defined judgement situation, according to the same software agent;
- a `MetaJudgement` situation to position the relevance assignment in the context of the case.

Metrics and measures computed on the projected concept graph then contribute to the definition of final relevance scores for concepts with respect to the analysed case. These relevance scores complement the initial relevance scores computed by software agents that implement text analytics pipelines and defined by human experts that relate concepts to the case. The scores can be once again represented following the approach described so far, i.e., by defining a judgement situation that relates a concept to a case and then a `RelevanceAssignmentOnJudgement` situation to quantify the relevance, according to the software agent that implements the concept relevance computation approach.

[30] https://www.w3.org/TR/swbp-n-aryRelations/#representation

The core ontology module for defining relevance thus provides a single, homogeneous framework that can be used to represent relevance scores resulting from the steps of the overall pipeline, from the initial relevance assessments made by experts that perform concept tagging for a case or computed by NLP pipelines that create links between concepts and the case at hand, to the final concept relevance scores resulting from the application of diffusion methods and projections over bipartite graph built from the initial relevance assignments.