Hybrid Intelligence Ontology Expansion and Insight Extraction

Mara Spadon[2688689], Laryza Mussavi[2687646], and Avani Maroo[2839671]

Vrije Universiteit Amsterdam, De Boelelaan 1105, 1081 HV, The Netherlands
Course: Knowledge Graphs and Semantic Technologies XM_0147
04-04-2025
Group 5

Abstract. Hybrid Intelligence (HI) combines the technical strengths of artificial agents with the cognitive capabilities of human intelligence, enabling more effective problem-solving through collaboration. Knowledge Graphs (KGs) serve as powerful tools to represent domain-specific information and integrate data from multiple external sources. This project aims to explore KGs in the context of HI by expanding an existing ontology and integrating data from relevant research papers. The project involves analyzing several HI-related papers, expanding the ontology, and extracting insights through data querying, KG metrics, and Machine Learning techniques. Finally, a research proposal will be introduced, addressing the challenges and limitations identified during the analysis, along with proposed solutions and an evaluation strategy.

Keywords: Hybrid Intelligence · Knowledge Graphs · Research Proposal

1 Introduction

Hybrid Intelligence (HI) is a field within Artificial Intelligence (AI) where the technical capabilities of artificial agents are combined with the cognitive capabilities of human intelligence. By leveraging the strengths of both human actors and artificial agents, HI enhances the ability to solve complex problems through collaboration. HI is particularly relevant in tasks such as decision-making, problem-solving, and smart monitoring, which can all by applied across a wide range of domains. Knowledge Graphs (KGs) are used to represent information about specific entities within a domain and to integrate data from multiple external sources in a coherent manner.

This project aims to explore the potential of KGs in the domain of HI by expanding an existing ontology and integrating data from various research papers to improve the understanding and capabilities of hybrid intelligence systems. This project will begin by analyzing several research papers from the field of HI and populating the ontology with relevant classes, subclasses, instances, and properties. Once the expanded ontology is established, valuable insights will be extracted by querying the data, analyzing KG measures, and applying Machine

Learning (ML) techniques. Finally, a research proposal will be presented, based on the challenges and limitations identified during the analysis of the expanded KG, along with a proposed solution and evaluation strategy. See Fig. 1 for a (provided) visualization of the original classes of the HI ontology and their instances.

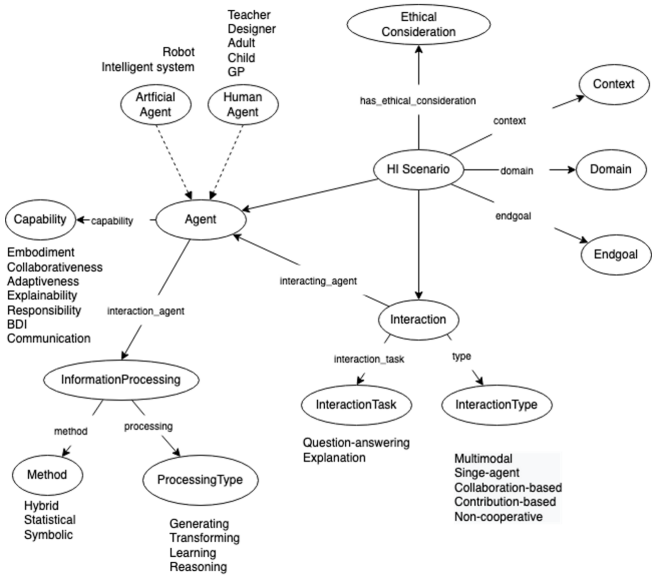

Fig. 1. Overview of original classes provided in the intial HI ontology with their instances

2 Data

2.1 Ontology Design and Expansion

For this project, the provided Hybrid Intelligence (HI) ontology was extended by systematically analyzing 18 research papers (6 papers per person) that intersected with (some parts) of the original HI ontology. The goal was to populate the existing classes of the ontology with instances found by scrutinizing the research papers one by one. Additionally, new classes and properties were incorporated in order to optimally enhance the expressiveness and utility of our version of the expanded ontology. This process, carried out by all the project groups within this course, culminated in the finalized expanded ontology. This refined ontology will serve as a foundation to gather insights in the subsequent section (see Section 3) and will be utilized to base our research proposal on (see Section 4). We will now outline all the modifications made to the original HI ontology.

2.1.1 New Classes

To better capture evaluation methodologies, we introduced an *EvaluationMethod* class, which enables explicit representation of various evaluation techniques applied to hybrid intelligence systems.

Furthermore, recognizing the growing emphasis on fairness in AI, we introduced *FairnessEvaluation* as a subclass of *EvaluationMethod*, ensuring that fairness-related evaluation techniques can be distinctly categorized.

Since various processing methodologies are frequently categorized into broader categories, we refined the ontology by introducing three subclasses under *ProcessingMethod*: *StatisticalMethod, SemanticMethod* and *MachineLearningMethod*.

Processing tasks can often be categorized in more general categories such as reasoning, transforming, generating, and learning, which also encompass a broad range of more specific tasks. Therefore we added the following classes as subclasses of the pre-existing class *ProcessingTask*: *ReasoningTask*, *TransformationTask*, *GenerationTask* and *LearningTask*.

The subclass *SocialAI* was added as a domain to help structure AI-related topcis that focus on ethical AI and fairness considerations, ensuring a clearer distinction between general AI domains and social AI.

The classes *Framework, Tool* and *Metric* were all added as new classes to the ontology. The *Framework* class defines any frameworks that were part of the research, such as any standardized frameworks or proposals of a new framework. The *Tool* class defines any tools that were used as part of the research. This may include online tools such as specialized software, or physical instruments. Lastly, the *Metric* class was added to describe any specific metrics used throughout a study where they based their analyses on.

HybridAgent and *DomainExpert* were added to the subclasses of *Agent*, namely *ArtificialAgent* and *Human*, respectively. The main motivation behind adding these classes was that these classes allow us to represent hybrid systems where artificial agents collaborate with human experts in domain-specific contexts.

Given the prevalence of knowledge graphs in the reviewed papers, we introduced the *KnowledgeGraphs* class as a subclass of *InformationProcessing* to emphasize their role in hybrid intelligence. One research highlighted two distinct types of knowledge graphs, namely the *DomainKnowledgeGraph* and the *CaseKnowledgeGraph*. For this reason, these two were added as subclasses to our newly defined class.

Below is a complete overview of all newly added classes, where indentations are used to indicate subclass relationships.

- *EvaluationMethod*
  - *FairnessEvaluation*
- *ProcessingMethod* (pre-existing class)
  - *StatisticalMethod*
  - *SemanticMethod*
  - *MachineLearningMethod*
- *ProcessingTask* (pre-existing class)

    - *ReasoningTask*
    - *TransformationTask*
    - *GenerationTask*
    - *LearningTask*
- *Domain* (pre-existing class)
    - *SocialAI*
- *Framework*
- *Tool*
- *Metric*
- *ArtificialAgent* (pre-existing class)
    - *HybridAgent*
- *Human* (pre-existing class)
    - *DomainExpert*
- *InformationProcessing* (pre-existing class)
    - *KnowledgeGraphs*
        * *DomainKnowledgeGraph*
        * *CaseKnowledgeGraph*

2.1.2 New Properties

See the overview below of the properties that were added to the original ontology together with their function and the motivation behind it.

- *collaboratesWith*: A symmetric property used to define that two actors can collaborate with one another. By defining both the domain and range as the class *Actor*, this property enables the explicit representation of reciprocal collaboration relationships within hybrid intelligence systems. This is especially useful for modeling interactions between human and artificial agents.
- *usesKG*: A transitive property that represents dependencies between knowledge graphs. If System A utilizes a knowledge graph from System B, and System C references the knowledge graph from System A, then by transitivity, System C indirectly uses the knowledge graph from System B.
- *evaluatedBy*: An object property that establishes a relationship between an artificial agent and the evaluation method used to assess its performance. Particularly useful for capturing evaluation frameworks referenced in research papers, ensuring that the ontology explicitly represents how different AI systems are assessed.
- *requiresHumanExpertise*: Many AI systems rely on human oversight, especially in Human-in-the-Loop AI paradigms. This object property links an AI system to the specific areas where human expertise is required, capturing essential dependencies between artificial agents and human decision-makers.
- *usesMetric*: An object property that connects an evaluation method to the metric it employs to measure performance. By explicitly linking methods to their respective metrics, this property helps ensure a structured and interpretable evaluation process.
- *followsFramework*: This property links a resource (e.g., an AI system, a research methodology, or a policy) to the framework it adheres to.
- *usesTool*: An object property that associates a resource with the specific tool it employs.

2.2 Instance Creation and Data Population

The instance creation process involved a manual extraction of relevant entities from the 18 research papers that were analyzed. These instances were added manually to ensure correctness and semantic consistency. Following the manual extraction, we leveraged generative AI tools, such as GPT-based models, to infer and generate additional property-instance connections based on the predefined domain and range constraints, which sped up the process significantly. Naturally, these generated connections were verified to ensure correctness. This approach allowed for a more comprehensive and systematically structured dataset while maintaining semantic accuracy.

2.3 Ontology Integration and Linking External Sources

By implementing ontology expansions, instance creations, and external integrations, we significantly enhanced the expressiveness and applicability of the HI Ontology, aligning it with contemporary hybrid intelligence research. In the end, we were able to expand the original ontology from 114 triples to 1,071 triples, and the number of nodes increased from 62 to 515. Visualizations of the impact on the original knowledge graph (Fig. 2) and the expanded ontology (Fig. 3) illustrate the increased complexity and interconnectivity, created using *Ontopea*[1].

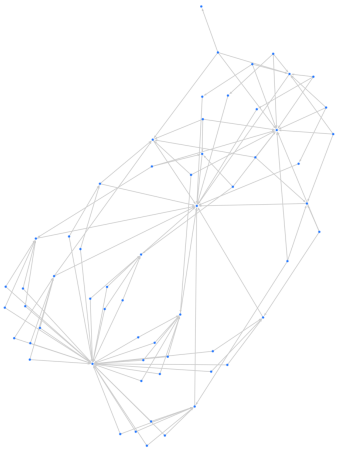

Fig. 2. Visualization of the original HI ontology, containing a total of 62 nodes

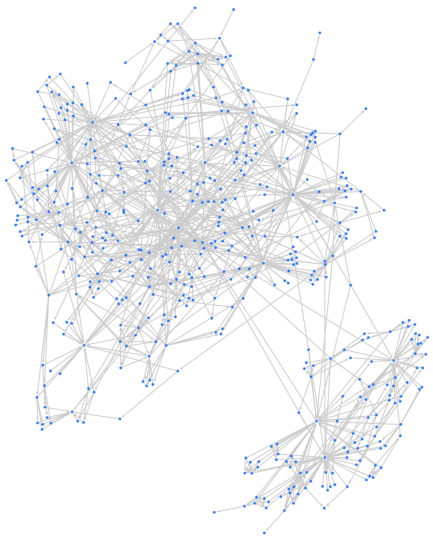

Fig. 3. Visualization of our expanded version of the HI ontology, containing a total of 515 nodes

[1] Access Ontopea here: `https://ontopea.com/`

While the exact connections may not be visible, the figures portray how our design choices contributed to a more extensive and interconnected structure. To enhance interoperability and connectivity with existing knowledge resources, we integrated our ontology with external sources such as DBpedia, Wikidata, and scholarly knowledge graphs (e.g., ORKG) using the `sameAs` function from the OWL library. This integration facilitates enriched knowledge discovery and enables linking entities within the HI Ontology to broader knowledge repositories. The linking process was performed by manually identifying relevant entities and aligning them with corresponding concepts in external sources.

3 Insights

In investigating the data and its characteristics, we employed various methods, including both visual and numerical summaries.[2] By examining the content of the ontology and the semantics of the relationships between its entities, the goal is to identify any anomalies that warrant further investigation and may inspire new approaches for processing and analyzing the data. This section presents the findings from the data analysis based on the HI ontology.

3.1 Numerical Summaries

To understand the structure of our knowledge graph, we converted the RDF data into a directed graph using NetworkX. We started by checking the graph for connectedness. Given the False result from this test, it is clear that our graph consists of multiple components that are not always reachable. Further insights into these components can be gained by providing an overview of the stronger and weaker (more or less connected) parts within the ontology. See Tables 1 and 2 for details.

Table 1. Strongly Connected Components (Sorted by Node Count)

Component	Nodes	Node Count
15	[n11dfb1e2c9904bbabc2e129f7c06b6c0b19, http://...	6
224	[http://www.semanticweb.org/vbr240/ontologies/...	4
757	[http://www.owl-ontologies.com/hi#AddPositiveF...	3
16	[http://www.semanticweb.org/vbr240/ontologies/...	3
597	[http://www.semanticweb.org/hi_ontology#Topogr...	2
⋮	⋮	⋮
1835	[Observation]	1
1836	[http://webprotege.stanford.edu/Rt8aWJ1SH2RxB3...	1
1837	[http://www.semanticweb.org/vbr240/ontologies/...	1
1838	[hasEvaluation]	1
1	[http://www.w3.org/2002/07/owl#Class]	1
[1871 rows x 3 columns]		

[2] All code is available at `https://github.com/LaryzaL/KGST`

Table 2. Weakly Connected Components (Sorted by Node Count)

Component	Nodes	Node Count
1	[http://www.semanticweb.org/vbr240/ontologies/...	1871
2	[http://www.semanticweb.org/hi_ontology#subCla...	8
3	[https://chowlk.linkeddata.es/, http://www.owl...	5

Analysis shows that there are many strongly connected components (1871), which are generally small, often consisting of only a few nodes or even individual nodes, with the largest containing just 6. This suggests that most entities do not have strong bidirectional links. In contrast, the largest weakly connected component includes the majority of nodes (1871), reflecting an overall weak, loosely connected structure where most entities remain indirectly reachable when edge direction is ignored. This pattern is not unexpected given how the HI ontology was constructed. Independently developed parts were later automatically merged into a single graph, contributing to both the fragmentation seen in the strongly connected components and the emergence of a large weakly connected structure.

The next thing we did was checking for blank nodes—entities without explicit URIs that often represent OWL constructs like restrictions or anonymous individuals. These can make querying and linking more challenging. In our case, we identified 22 blank nodes, a reasonable number that suggests some abstraction but not excessive ambiguity.

Next, we calculated basic statistics about the graph:

- **Nodes:** 1884
- **Edges:** 3850
- **Density:** 0.0011

The low density reflects the expected sparse structure of ontologies, where most entities have only a few semantic connections.

To gain a foundational understanding of the ontology's structure, we extracted key metrics using two complementary methods: SPARQL queries and Protégé's built-in metrics dashboard. The SPARQL-based analysis revealed approximately 131 classes, 620 individuals, and 81 properties, resulting in a total of 3,861 RDF triples and 1,275 unique entities. Protégé provided a slightly different snapshot, reporting 137 classes, 1,032 individuals, and 85 object properties, along with detailed axiom counts. The discrepancies arise due to differences in how Protégé and SPARQL interpret blank nodes, annotation properties, and class declarations—Protégé includes inferred structures and annotations by default, whereas SPARQL only retrieves explicitly asserted triples. Nevertheless, both views offer consistent insights into the graph's size and complexity, confirming a rich and reasonably populated ontology.

3.2 Visual Summaries

We examined how connected nodes are by analyzing their degree (number of

edges). The mean degree was 4.08, and the mean degree centrality was 0.0021—confirming that most nodes are only lightly connected. The degree distribution was highly skewed, with a few nodes having many connections. These high-degree nodes likely serve as semantic hubs, a common feature in real-world knowledge graphs.

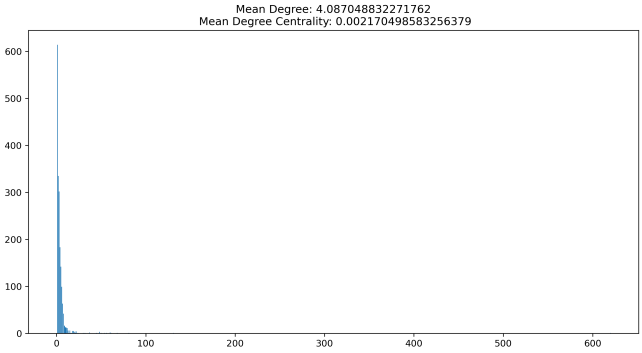

Fig. 4. Mean Degree Centrality

To better visualize this pattern, we applied different plotting strategies: one log-scaled x-axis helped reveal the long tail of rare, high-degree nodes (see Fig. 5). Then, we grouped degrees into logarithmic bins, summarizing the spread more clearly and smoothing out noisy degree-to-degree fluctuations. This showed that most nodes had between 2 and 6 connections, while a few had hundreds—again highlighting hub-like behavior (see Fig. 6).

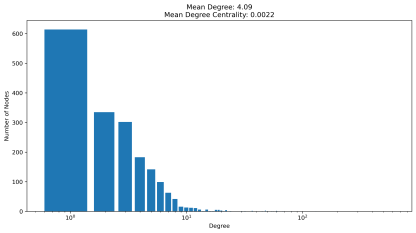

Fig. 5. Log-Scaled Degree Centrality

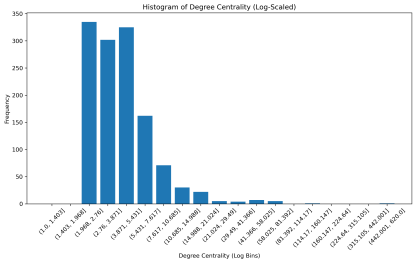

Fig. 6. Logarithmic Bins - Degree Centrality

We then calculated closeness centrality, which measures how quickly a node can reach all others in the graph. The average value was 0.0020, indicating that the graph is loosely connected, with many nodes being far from others. Three

plots helped us analyze this further: One linear histogram confirmed that most nodes have very low closeness scores (see Fig. 7). One version with a logarithmic x-axis helped us see subtle variations that were otherwise compressed (see Fig. 8). Finally, we grouped the scores into logarithmic bins with labeled ranges. This version made it clear that a large majority of nodes are in the lowest ranges of reachability, while only a few nodes have significantly higher centrality. These nodes likely play important roles in connecting different parts of the graph (see Fig 9).

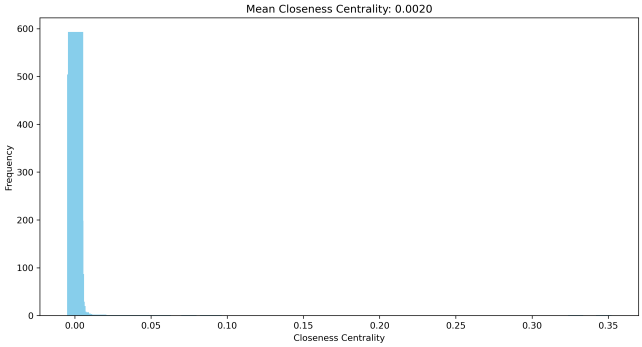

Fig. 7. Mean Closeness Centrality

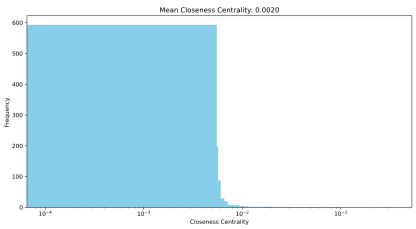

Fig. 8. Log-Scaled Closeness Centrality

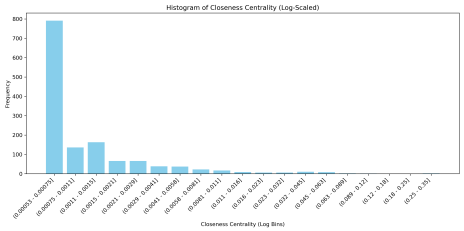

Fig. 9. Logarithmic Bins - Closeness Centrality

To better understand the structure of our knowledge graph, we generated a simplified version that includes only individuals and their object property assertions, excluding annotation and literal assertions (e.g., labels). This was done to reduce visual clutter and enhance interpretability.

The simplified graph — interactive graph available via GitHub — offers a clearer overview of instance-level relationships (see Fig. 10). To further analyze modularity, we applied Louvain community detection. Figure 19 (see Appendix

A.2) shows the communities plotted separately, revealing meaningful clusters of individuals. Interestingly, the community split aligns well with what one might intuitively expect based on the interactive graph, highlighting a consistent modular structure within our ontology.

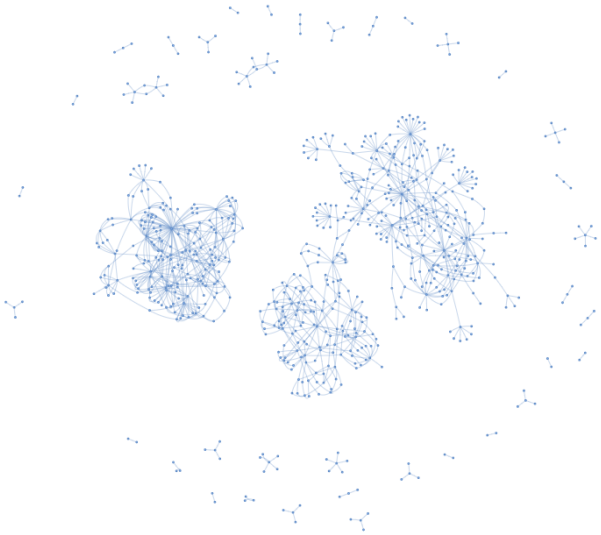

Fig. 10. Simplified HI KG

Querying using SPARQL To extract meaningful insights from our knowledge graph, we designed and executed a set of SPARQL queries, ranging from basic statistics to complex structural patterns. Each query highlights different aspects of the graph's semantics and structure.

1. Most Frequently Referenced Individuals
This query counts how often each individual appears as a subject or object in any triple. It helps identify central entities—those most frequently referenced across the knowledge graph. Fig. 11 shows that most individuals are referenced only a few times, while a small number appear frequently—indicating a highly skewed distribution with a few central entities.

2. Most Frequently Used Predicates
We retrieved the most commonly used properties (predicates) to understand how relationships are formed within the graph. The query reveals which predicates structure the majority of the graph, offering insight into the ontology's modeling style. Fig.12 shows that a small set of core predicates—including rdf:type, rdfs:label, and hasDescription—are used heavily, suggesting a consistent modeling strategy with a central semantic vocabulary.

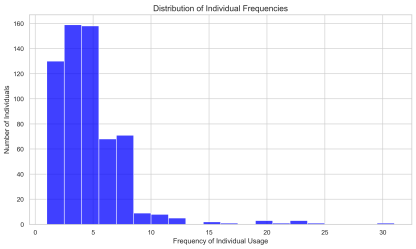

Fig. 11. Distribution of individual reference frequencies

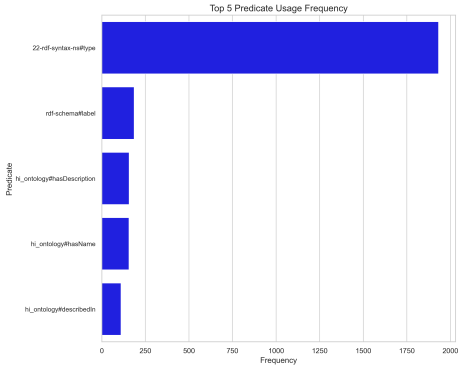

Fig. 12. Top 5 most frequently used predicates

3. Instances per Class

This query counts how many individuals belong to each class. It provides a sense of how densely populated each conceptual category is and whether some classes are underrepresented or overused in the data. Fig. 13 shows a long-tail distribution where most classes contain only a few instances, indicating sparse population across much of the ontology.

4. Individuals by Predicate Diversity

This query ranks individuals by the number of distinct relationships (predicates) they are involved in — either as a subject or object. It identifies semantically rich entities that connect to many different types of information, beyond just frequency. Fig. 14 highlights the most semantically diverse individuals—those connected through many different types of relationships, not just frequently but across varied contexts.

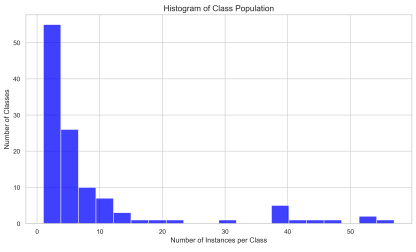

Fig. 13. Distribution of the number of instances per class

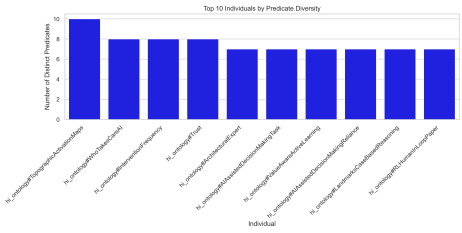

Fig. 14. Top individuals ranked by the number of distinct predicates they are involved in

5. Most Frequent Class-to-Class Connections
This query finds the most common class-to-class links, by examining how instances of one class relate to instances of another. It helps uncover dominant structural patterns in the graph, such as whether certain classes frequently co-occur or interact. Fig. 15 highlights how different classes in the ontology are interconnected.

6. Classes Where Individuals Are Interlinked
We queried classes where individuals of the same class link to one another through some property. This can signal internal cohesion within a class and highlight collaborative or hierarchical structures modeled in the ontology. Fig. 16 highlights which classes contain individuals that frequently connect to each other.

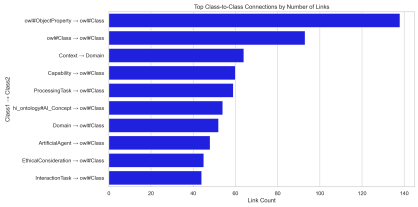

Fig. 15. Top class-to-class connections by frequency

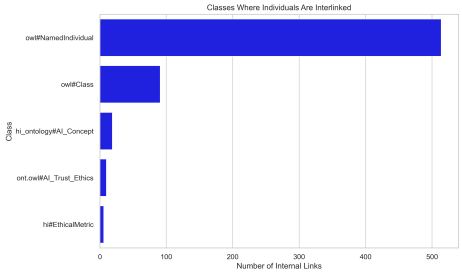

Fig. 16. Classes where individuals are frequently linked to other individuals of the same class

3.3 ML Methods

To explore the potential for automatic knowledge graph enrichment, we applied link prediction—a task that aims to infer missing relationships between entities based on existing structure. This allows us to assess whether the graph encodes meaningful structural signals and whether its connectivity patterns are predictive.

To perform link prediction, we trained a Random Forest classifier using structural features computed for both positive (existing) and negative (non-existing) links. The dataset included a total of 6420 samples, split into training and testing sets, wherein 30% of the data went into the test set, and the remaining 70% was used for training. For each candidate link, we extracted features such as common neighbors, Jaccard coefficient, Adamic-Adar index, and preferential attachment, which capture local structural information between nodes.

While the classifier demonstrated strong performance overall, with an accuracy of 81.5% and an AUC of 0.90 (see Fig. 18), the confusion matrix revealed a moderate number of false negatives—cases where true links were missed (see

Fig. 17). This suggests that while local graph structure captures many useful patterns, it may not fully represent the semantic complexity of relationships in the ontology.

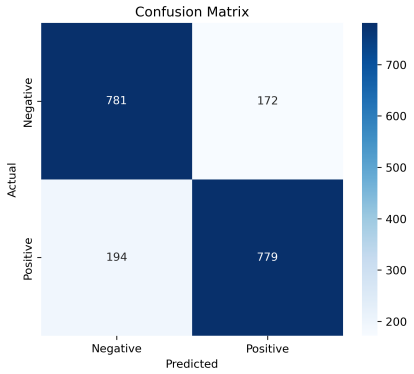

Fig. 17. Confusion Matrix

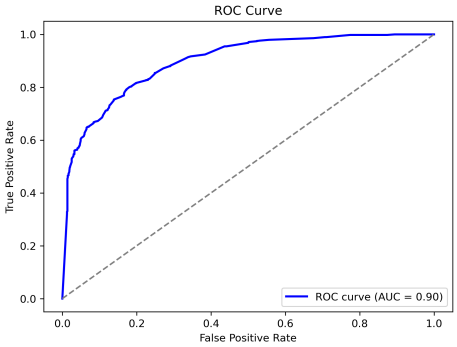

Fig. 18. AUC-ROC

4 Research Proposal

4.1 Problem Definition

Given the insights from the previous section, several problems with the HI knowledge graph have been identified. One of the primary issues is that the graph is sparse, containing many weakly connected components, as confirmed by the visualization and the low degree and closeness centrality values of the nodes. These limited connections could hinder the reuse and interoperability of the ontology, undermining some of the core purposes of linked data. Additionally, the lack of modularity is evident in the Louvain communities, which are inconsistently partitioned into many small groups, making it difficult to efficiently inspect different parts of the graph. Furthermore, blank nodes exacerbate these issues, making it harder to interpret and understand the relationships and entities due to their lack of identifiers. Without proper reduction, these blank nodes introduce additional ambiguity into an already sparse graph. They also add noise, which may hinder the generation of accurate embeddings, ultimately affecting tasks like link prediction or clustering. Finally, blank nodes increase computational complexity, which is especially problematic when using resource-intensive methods for graph analysis. Overall, the current structure of the ontology is too disorganized to effectively derive meaningful inferences or integrate with other sources.

4.2 Motivation

Although blank node reduction, ontology partitioning, and link prediction are often studied and applied separately, they address similar goals and limitations, suggesting that integrating them could offer complementary benefits. They all tackle issues related to scalability, sparsity, and incompleteness, which are common characteristics of dynamic, real-world data. Companies and institutions that work with such data and manage parts of their data using knowledge graphs, such as Uber and TNO, would benefit from this approach. By requiring fewer computational resources, it could save both time and costs. Additionally, this proposal would be valuable to researchers working in the fields of semantic technologies, ontology engineering, and knowledge graph optimization. It could help them clean and restructure their data, thus improving knowledge discovery and opening up new opportunities to explore more combined approaches. We believe this solution could optimize how data is stored in knowledge graphs, ultimately ensuring smoother workflows that require less manual effort while also being more computationally efficient.

4.3 Related Work

4.3.1 Ontology Decomposition

Ontology decomposition involves breaking down large ontologies into smaller, manageable modules while preserving their semantics. This is essential in large-scale knowledge graphs, where tasks like reasoning, alignment, and querying become increasingly complex.

Many existing approaches to ontology decomposition focus on modularization — breaking down large ontologies into smaller, easier-to-manage parts. Some techniques aim to improve scalability by using parallel reasoning for specific logical fragments like EL [1]. Others use atomic decomposition to find small, overlapping pieces of the ontology that help organize and visualize its structure [2]. There are also graph-based methods that split the ontology into semantically meaningful subgraphs to make it easier to understand and check for consistency [3]. Researchers have compared different types of modularization — syntactic, semantic, and hybrid — to see how well they support tasks like reuse, alignment, and reasoning [4]. In fields like biomedicine, targeted extraction of ontology subsets has been useful for improving scalability and making it easier to annotate data across different domains [5].

Despite these advances, most approaches remain symbolic and rule-based. The use of machine learning models for ontology decomposition is still in its early stages. [6]. Exploring these methods offers a promising direction for developing adaptive, context-aware decomposition frameworks that scale more effectively across complex knowledge graphs.

4.3.2 Blank Node Reduction

In RDF, a blank node (or Bnode) is a node in an RDF graph without a URI or literal identifier [7]. Blank nodes are commonly used to connect complex values

without adding extra nodes. However, their presence complicates data merging across sources, discouraging their use in knowledge graphs [8]. Excessive use of blank nodes can hinder SPARQL query efficiency, requiring special handling and making queries more complex. One recent study proposes the *HERSE* framework that leverages Skolemization to replace blank nodes with unique URIs, enhancing RDF dataset interoperability [9]. However, it lacks any context-aware techniques or dynamic approaches when real-time updates are necessary. Other studies utilize embeddings [10] or rough set theory [11] to match blank nodes while merging two graphs. As our final dataset was constructed by merging various knowledge graphs, these approaches could be considered for the reduction of blank nodes in our graph.

4.3.3 Link Prediction for Knowledge Graph Refinement

Ontology alignment identifies overlapping entities and relations for meaningful data exchange. Automating this task often results in unstructured KGs with misalignments and missing relationships. Link prediction (LP), which infers missing facts from existing KG data, offers a solution [12]. Machine learning methods, including KG embeddings and Large Language Models (LLMs), help detect missing links and entities while reducing noise, improving connectivity.

Link prediction using KG embeddings transforms KGs into low-dimensional representations to infer new facts. Popular basic models like TransE [13], DistMult [14], ComplEx [15] struggle as KGs grow larger and noisier, particularly with real-world data [16]. These models also tend to over-represent well-connected entities and neglect sparse nodes due to insufficient data [12].

LLMs have gained attention in LP due to their ability to process textual data and extract semantic relationships, addressing scalability and connectivity issues in embedding-based models. LLM_sim [17], for example, reduces noise and enhances graph completeness by applying a contextual approach. A fuzzy search retrieves similar triples, which are replaced with LLM-generated candidates based on similarity scores relative to the original KG. Similar frameworks [18,19] have, like LLM_sim, proven effective in noise reduction and KG completion. However, these methods mainly focus on entity alignment and refining triples, often neglecting other structural issues that induce noise, such as lack of modularity, sparsity, and dense clusters that add complexity.

4.4 Research Hypothesis and RQ

Our research question can be formulated as follows:

"How can LLMs and ML-based methods be used to automatically decompose large, complex ontologies into structured sub-ontologies while reducing blank nodes and optimizing hierarchical consistency using link prediction?"

To answer this research question, we consider two hypotheses:

- *Hypothesis 1*: "Integrating blank node reduction and ontology partitioning significantly improves the completeness and accuracy of a knowledge graph, while also reducing computational complexity through more efficient graph structure."
- *Hypothesis 2*: "The application of LLM-based methods for link prediction significantly enhances the accuracy of inferred triples and improves the completeness of knowledge graphs."

4.5 Proposed Approach and Contribution

We propose a hybrid framework that combines Bnode reduction, graph decomposition, and link prediction in large knowledge graphs using machine learning and LLMs. The pipeline consists of the following stages:

1. **Context-Aware Blank Node Management**: This stage aims to enhance efficiency and reduce computational time by pruning low-information triples while preserving semantic richness, achieved through embedding similarity (nearly identical blank nodes may be merged or removed) and signature-based methods (assigning identifiers based on the structural roles of entities).
2. **Ontology Partitioning**: Following Bnode reduction, a clustering method, such as the K-means variant described in [6], can be applied using similarity scores based on graph-based structural dependencies (e.g., subclass hierarchies) to divide the ontology into meaningful modules.
3. **Link Prediction with LLM**: In this stage, we introduce an LLM to perform link prediction. Methods like LLM_sim are employed to handle noise and generate missing triples, facilitating knowledge graph completion.
4. **Validation**: Finally, a link prediction model like ComplEx is used to validate the inferred triples and integrate them into the knowledge graph, ensuring their accuracy and consistency.

This framework addresses scalability and connectivity challenges in large, complex KGs. It offers a novel solution that combines KG reorganization with advanced link prediction for context-aware knowledge graph refinement.

4.6 Experimental Evaluation

To evaluate the proposed approach, we will use a state-of-the-art LP model like ComplEx, which is more suitable as it overcomes the limitations of models like TransE and DistMult in handling diverse relation patterns such as asymmetry and one to many mappings [20]. Key evaluation metrics, including Mean Reciprocal Rank (MRR), Hits@K, precision, recall, and F1-score, would then be obtained to assess improvements in KG completeness and alignment accuracy. For Hypothesis 1, these metrics will be obtained and evaluated after Bnode reduction and ontology partitioning (stages 1 and 2 of the pipeline), comparing the results to the original KG. For Hypothesis 2, the evaluation takes place after the

full pipeline, comparing performance to both the original KG and the intermediate results from Hypothesis 1 to assess the added value of the LLM-based link prediction. Additionally, structural changes such as community distribution and data sparsity will be analyzed as in Section 3, and the method will be validated on other benchmark datasets like WN18RR [21] and FB15k-237 [22] to ensure robustness and generalizability.

5 Conclusion

In this project, we explored the development, enrichment, and analysis of a domain-specific knowledge graph for hybrid intelligence applications. Beginning with ontology construction, we curated and integrated concepts from multiple sources, using both manual effort and generative language models to enhance semantic richness.

We then examined the graph structure through numerical and visual summaries, revealing its sparse connectivity, modularity, and the presence of central semantic hubs. To gain deeper insights, we executed SPARQL queries—ranging from basic summaries to more complex, each revealing meaningful trends in class usage, predicate diversity, and inter-entity relationships. These symbolic analyses were complemented by a supervised link prediction model, which identified potential missing edges with promising accuracy and AUC scores. Together, these methods demonstrated how hybrid techniques can uncover patterns and validate ontology structure.

Finally, we proposed a novel research direction involving ontology decomposition, blank node reduction, and LLM-based link prediction. This integrated approach aims to automate sub-ontology extraction while preserving semantic consistency and improving scalability. Overall, the project highlights how structured reasoning and machine learning can work together to extract, validate, and enhance knowledge in complex semantic graphs.

References

1. Y. Kazakov, M. Krötzsch, and F. Simančík, "Concurrent Classification of EL Ontologies," *The Semantic Web* – ISWC 2011, L. Aroyo et al., Eds. Berlin, Heidelberg: Springer, 2011, vol. 7031, pp. 305–320.
2. C. Del Vescovo, B. Parsia, U. Sattler, and T. Schneider, "The modular structure of an ontology: Atomic decomposition," *Proc. 22nd Int. Joint Conf. Artif. Intell. (IJCAI)*, Barcelona, Spain, 2011, pp. 2232–2237. [Online].
3. J. M. Taylor and V. Raskin, "Graph decomposition and its use for ontology verification and semantic representation," \textit{Proc. Int. Conf. on Artificial Intelligence (ICAI)}, Las Vegas, NV, USA, 2011, pp. 716–722.[Online].
4. M. d'Aquin, A. Schlicht, H. Stuckenschmidt, and M. Sabou, "Ontology modularization for knowledge selection: Experiments and evaluations," \textit{Proc. 18th Int. Conf. Database and Expert Systems Applications (DEXA)}, Regensburg, Germany, 2007, pp. 874–883. [Online].
5. O. Bodenreider, C. Tao, and J. Jiang, "Cross-domain targeted ontology subsets for annotation: The case of SNOMED CT," *J. Biomed. Inform.*, vol. 46, no. 4, pp. 727–733, Aug. 2013. [Online].
6. S. S. Ahmed, M. Malki, and S. M. Benslimane, "Ontology Partitioning: Clustering Based Approach," *International Journal of Information Technology and Computer Science*, vol. 7, no. 6, pp. 1–11, 2015, doi: 10.5815/ijitcs.2015.06.01.
7. Y. Tzitzikas, C. Lantzaki, and D. Zeginis, "Blank Node Matching and RDF/S Comparison Functions," *Lecture Notes in Computer Science*, vol. 7649, pp. 591–607, 2012, doi: https://doi.org/10.1007/978-3-642-35176-1_37.
8. A. Hogan, M. Arenas, A. Mallea, and A. Polleres, "Everything You Always Wanted to Know About Blank Nodes," *SSRN Electronic Journal*, 2014, doi: https://doi.org/10.2139/ssrn.3199109.
9. A. Beldi, S. Sassi, R. Chbeir, and Abderrazek Jemai, "HERSE: Handling and Enhancing RDF Summarization Through Blank Node Elimination," *Lecture notes in computer science*, vol. 14670, pp. 87–101, Jan. 2024, doi: https://doi.org/10.1007/978-3-031-62700-2_9.
10. A. Becker, M. A. Sherif, and A.-C. Ngonga Ngomo, "Blink: Blank Node Matching Using Embeddings," *Lecture Notes in Computer Science*, vol. 15231, pp. 218–236, Nov. 2024, doi: https://doi.org/10.1007/978-3-031-77844-5_12.
11. J. de Almeida Monte-Mor and A. M. da Cunha, "ApproxMap - a method for mapping blank nodes in RDF datasets," *Journal of the Brazilian Computer Society*, vol. 21, no. 1, Apr. 2015, doi: https://doi.org/10.1186/s13173-015-0022-3.
12. A. Rossi, D. Barbosa, D. Firmani, A. Matinata, and P. Merialdo, "Knowledge graph embedding for link prediction: A comparative analysis," *ACM Transactions on Knowledge Discovery from Data (TKDD)*, vol. 15, no. 2, pp. 1-49, 2021.
13. A. Bordes, N. Usunier, A. Garcia-Duran, J. Weston, and O. Yakhnenko, "Translating embeddings for modeling multi-relational data," *Advances in Neural Information Processing Systems*, vol. 26, 2013.
14. B. Yang, W. T. Yih, X. He, J. Gao, and L. Deng, "Embedding entities and relations for learning and inference in knowledge bases," *arXiv preprint arXiv:1412.6575*, 2014.
15. T. Trouillon, J. Welbl, S. Riedel, É. Gaussier, and G. Bouchard, "Complex embeddings for simple link prediction," *International Conference on Machine Learning*, pp. 2071-2080, Jun. 2016.

16. I. Ferrari, G. Frisoni, P. Italiani, G. Moro, and C. Sartori, "Comprehensive analysis of knowledge graph embedding techniques benchmarked on link prediction," *Electronics*, vol. 11, no. 23, p. 3866, 2022.
17. N. Dong, N. Kertkeidkachorn, X. Liu, and K. Shirai, "Refining Noisy Knowledge Graph with Large Language Models," *Proceedings of the Workshop on Generative AI and Knowledge Graphs (GenAIK)*, pp. 78-86, Jan. 2025.
18. L. Yao, J. Peng, C. Mao, and Y. Luo, "Exploring large language models for knowledge graph completion," *ICASSP 2025-2025 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, pp. 1-5, Apr. 2025.
19. L. Yang, H. Chen, X. Wang, J. Yang, F. Y. Wang, and H. Liu, "Two heads are better than one: Integrating knowledge from knowledge graphs and large language models for entity alignment," *arXiv preprint arXiv:2401.16960*, 2024.
20. S. Bonner, I. P. Barrett, C. Ye, R. Swiers, O. Engkvist, C. T. Hoyt, and W. L. Hamilton, "Understanding the performance of knowledge graph embeddings in drug discovery," *Artificial Intelligence in the Life Sciences*, vol. 2, p. 100036, 2022.
21. "Papers with Code - WN18RR Dataset," *Paperswithcode*, 2022. [Online]. Available: https://paperswithcode.com/dataset/wn18rr. [Accessed: Apr. 04, 2025].
22. "Papers with Code - FB15k-237 Dataset," *Paperswithcode*. [Online]. Available: https://paperswithcode.com/dataset/fb15k-237. [Accessed: Apr. 04, 2025].

A Appendix

A.1 Individual Contributions

Mara Spadon Analyzed 6 papers to populate original ontology, wrote the abstract and sections 1, 2 and 4.3.2 of the report. Generated a jupyter notebook containing the ML approach to apply link prediction, discussed in section 3.3. Proofread the entire report before handing it in.

Laryza Mussavi Analyzed 6 papers to populate original ontology, contributed to section 3.1, 3.2 and 4 of the report excluding 4.3.1 and 4.3.2. Generated a jupyter notebook containing numerical and visual summaries of the ontology, discussed in section 3.1 and 3.2. Contributed to part of the SPARQL Queries notebook, discussed in section 3.2. Proofread the entire report before handing it in.

Avani Maroo Analyzed 6 papers to populate original ontology, wrote section 3, 4.3.1 and the conclusion of the report. Generated a jupyter notebook for SPARQL queries discussed in section 3.2 and also contributed to the numerical and visual summaries jupyter notebook. Proofread the entire report before handing it in.

A.2 Louvain Communities

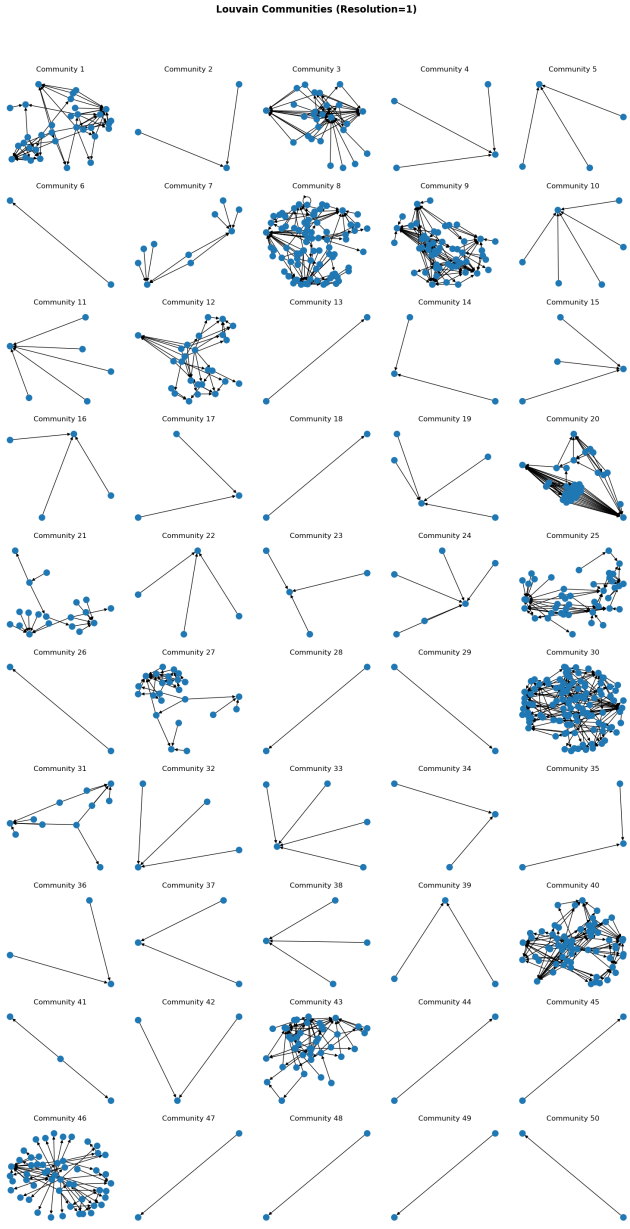

Fig. 19. Appendix A.2: Louvain Communities