Towards the Automation of Knowledge Graph Construction Using Large Language Models

Vamsi Krishna Kommineni[1,2,3,*], Birgitta König-Ries[1,2,4] and Sheeba Samuel[5]

[1]Heinz Nixdorf Chair for Distributed Information Systems, Friedrich Schiller University Jena, Jena, Germany
[2]German Centre for Integrative Biodiversity Research (iDiv) Halle-Jena-Leipzig, Leipzig, Germany
[3]Max Planck Institute for Biogeochemistry, Jena, Germany
[4]Michael Stifel Center Jena, Jena, Germany
[5]Distributed and Self-organizing Systems, Chemnitz University of Technology, Germany

Abstract

The conventional process of building Ontologies and Knowledge Graphs (KGs) heavily relies on human domain experts to define entities and relationship types, establish hierarchies, maintain relevance to the domain, fill the ABox (i.e., populate with instances), and ensure data quality (including amongst others accuracy and completeness). On the other hand, Large Language Models (LLMs) have recently gained popularity for their ability to understand and generate human-like natural language, offering promising ways to automate aspects of this process. This work explores the (semi-)automatic construction of KGs facilitated by different state-of-the-art LLMs: Mixtral 8x22B Instruct v0.1, GPT-4o, GPT-3.5, and Gemini. Our pipeline involves formulating competency questions (CQs), developing an ontology (TBox) based on these CQs, constructing KGs using the developed ontology, and evaluating the resultant KG with minimal to no involvement of human experts. We showcase the feasibility of our semi-automated pipeline by creating a KG on deep learning methodologies by exploiting scholarly publications. The answers generated via Retrieval-Augmented-Generation (RAG) were evaluated by a domain expert manually, and the KG was evaluated by matching the KG individuals to RAG-generated answers. Our findings suggest that employing LLMs could potentially reduce the human effort involved in the construction of KGs, although a human-in-the-loop approach is recommended to evaluate automatically generated KGs.

Keywords

Knowledge Graphs, Ontology, Competency Questions, Large Language Models, Retrieval-augmented generation

1. Introduction

In information organization and representation, ontologies stand as foundational frameworks for describing and structuring domain knowledge. These structured representations not only represent the entities and relationships within a domain but also serve as the foundation for constructing comprehensive knowledge graphs (KGs) [1]. KGs, in turn, offer a powerful mechanism for interlinking diverse pieces of information and facilitating sophisticated data analytics and reasoning. The domain knowledge encapsulated within ontologies constitutes a valuable asset for various knowledge-intensive applications. This comes at a price, though: ontology and knowledge engineering represents a collaborative and interdisciplinary effort, demanding the time and expertise of multiple stakeholders [2, 3]. The higher the expressiveness in the ontology language, the more intricate design choices need to be made throughout the construction process, with additional developmental challenges, including accuracy, scalability, and depth of knowledge captured. The conventional approach to constructing KGs typically involves gathering domain requirements through CQs (also proposed as the requirement specification in ontology development) collected from domain experts, collaborating with computer scientists and domain specialists to develop an ontology, transforming unstructured data into structured formats, and finally, populating the ontology to create the KG [4]. In this context, a pressing question emerges: How

NLP4KGC: 3rd International Workshop on Natural Language Processing for Knowledge Graph Creation, September 17, 2024, Amsterdam, Netherlands.

*Corresponding author.
vamsi.krishna.kommineni@uni-jena.de (V. K. Kommineni); birgitta.koenig-ries@uni-jena.de (B. König-Ries); sheeba.samuel@informatik.tu-chemnitz.de (S. Samuel)
0000-0001-6168-3085 (V. K. Kommineni); 0000-0002-2382-9722 (B. König-Ries); 0000-0002-7981-8504 (S. Samuel)

can we strike a balance between the resource-intensive nature of knowledge graph construction and the need to automate the process effectively?

This research paper attempts to address this question by leveraging Large Language Models (LLMs) in Knowledge Graph Engineering, particularly focusing on minimizing the time and human effort involved in these processes while maintaining the level of quality achieved by traditional approaches. In recent years, the emergence of these models has revolutionized the landscape of natural language processing (NLP) and knowledge representation. Equipped with massive pre-trained parameters and advanced neural architectures, LLMs exhibit remarkable capabilities in understanding and generating human-like text across a spectrum of languages and domains [5, 6]. As a result, they have gained immense popularity and adoption across diverse fields ranging from information retrieval to language translation.

Hence, we explore the (semi-)automatic construction of a KG, starting from collecting competency questions (CQs) to creating an ontology and to filling the data in the ontology, facilitated by the integration of LLMs. Building upon our previous work [7], which introduced a six-component pipeline encompassing data collection, competency question (CQ) generation, ontology creation, CQ answering, KG construction, and evaluation (Figure 1), we now focus on refining the ontology and KG construction phase and its evaluation by incorporating a wider range of state-of-the-art LLMs (Mixtral 8x22B Instruct v0.1[1], GPT-4o[2], GPT-3.5[3], and Gemini[4]) and expanding the dataset. To test the feasibility of our approach, we apply this methodology to creating an ontology and KGs about deep learning (DL) methodologies. This involved extracting essential information required for reproducibility from scholarly publications employing DL in the biodiversity domain.

The choice of this example has been motivated by the increasing usage of DL in research. Documenting the provenance of DL results is indispensable to facilitate the reproducibility of these studies, a prerequisite to trust and validation of results. To accomplish this task effectively, information, such as models, architectures, hyperparameters, and other key details, needs to be captured and stored in a structured representation.

2. Related work

LLMs have revolutionized knowledge engineering and NLP, showcasing human-level performance across diverse linguistic tasks [5, 6]. With the increasing robustness of LLMs, their potential as a knowledge source in various applications such as KG completion, ontology refinement, and question answering has become evident [8, 9, 10]. Research has rapidly expanded to explore the application of LLMs, with recent papers providing surveys on the use of LLMs in KG engineering along with associated challenges [11, 3]. Meyer et al. [11] presents a list of application areas for LLM-assisted KG engineering, including creating or enriching KG schemas/ontologies, populating KGs, etc. In their position paper, Pan et al. [3] present opportunities, visions, research topics, and challenges for LLMs for KGs and KGs for LLMs. Despite the potential of LLMs, they have shown limited success in generating Competency Questions (CQs), with generally low precision scores and inconsistent improvements when using specific prompting techniques [12].

Recent studies have introduced methods for ontology creation [13, 2], augmentation [14], completion [15], and learning [16] using LLMs. Cohen et al. [13] present a crawling approach to extract a KG from an LLM using 'subject-relation-object' statement formats. Funk et al. [2] focus on constructing a concept hierarchy for a given domain starting from a seed concept by querying LLMs. However, they consider only subconcept/is-a relation, but no other relations. In this work, we focus on reusing existing ontologies with all their concepts and relations. While there is limited research on the utilization of LLMs for completing KGs or ontologies [17], this trend appears to be changing rapidly. While

[1] https://huggingface.co/mistralai/Mixtral-8x22B-Instruct-v0.1
[2] https://openai.com/index/hello-gpt-4o/
[3] https://platform.openai.com/docs/models/gpt-3-5
[4] https://gemini.google.com/

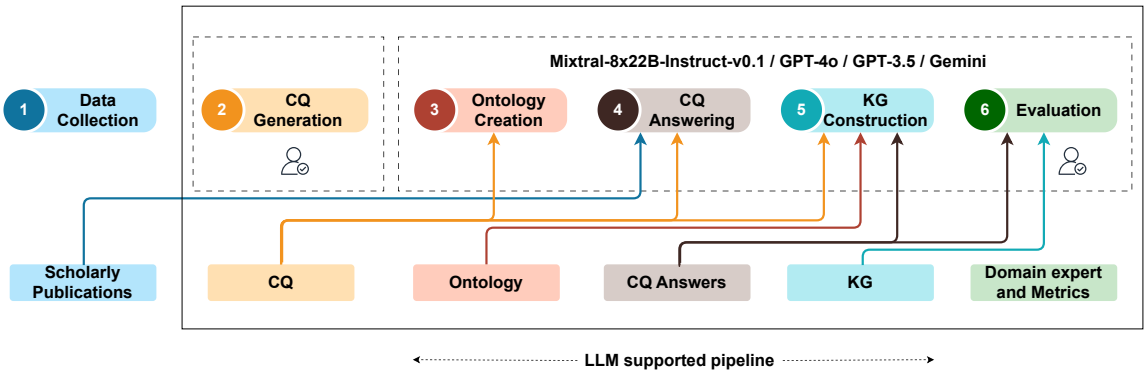

Figure 1: The (semi-)automatic approach for constructing KGs. The second and sixth stages of the LLM-supported pipeline allow human input.

LLMs offer new possibilities for ontology learning and development, they do not alter the fundamental need for expert collaboration and establishing consensus within a community. However, they may enhance the productivity of ontologists by simplifying ontology development and integrating into semi-automatic toolchains [18]. In this work, we have reused the existing ontology in the domain and involved humans in the loop to validate and evaluate the LLM-generated content. Several studies have proposed ontologies within the machine learning domain [19, 20, 21, 22, 23]. While domain experts construct some of these, they often lack accompanying lists of competency questions and specifications for ontology requirements. In contrast, the current approach in this work is guided by competency questions generated by Large Language Models (LLMs), aimed at describing the methodologies in deep learning necessary for their reproducibility.

Most of the publications discussed above [2, 11, 18, 15, 14] are based on openAI's GPT 3.5[5] or 4[6]. However, the usage cost of OpenAI API models can escalate quickly if deployed for large-scale applications. In contrast, open-source LLMs offer transparency, model control, usage flexibility, and cost-effectiveness. As far as our knowledge extends, this represents the first approach to introduce a comprehensive (semi-)automated pipeline for constructing ontology and KGs using state-of-the-art closed and open-source LLMs. Furthermore, we demonstrate and assess each stage of the pipeline using real-world examples extracted from biodiversity publications using DL methodologies.

3. Methods

This section outlines the current approach to constructing knowledge graphs utilizing large language models. We briefly mention each step of our pipeline and provide more details on the KG population using different LLMs.

To facilitate transparency and encourage reproducibility, the prompts, code, generated results, and the underlying data employed in the evaluation process are all publicly accessible on GitHub: https://github.com/fusion-jena/automatic-KG-creation-with-LLM.

3.1. Data Collection

We reused a dataset generated in previous research [24, 25], where authors conducted a systematic literature review to identify publications employing DL methods in biodiversity research based on keywords suggested by biodiversity experts [26]. Additionally, two domain experts curated a dataset of 61 publications and manually extracted reproducibility-related variables [25] [7]. These variables,

[5] https://platform.openai.com/docs/models/gpt-3-5
[6] https://platform.openai.com/docs/models/gpt-4
[7] The dataset with reproducibility-related variables is available at https://github.com/fusion-jena/automatic-KG-creation-with-LLM/blob/master/Data/Publication_metadata.csv

inspired by the current literature [27, 28, 29], serve as the basis for assessing the reproducibility of DL methods and are essential for ensuring method reproducibility. We used 30 out of 61 publications in this work to test the (semi-) automated pipeline.

3.2. CQ generation

Given the critical role of competency question (CQ) generation in the overall pipeline and also the lack of LLM capabilities[12], in this step, two domain experts (the first and last authors of this publication) formulated a set of 28 CQs[8] that cover important aspects of the deep learning pipeline, starting from the data collection to the deployment of the DL model, considering the current literature and recommendations [27, 28, 25]. Each CQ is also now accompanied by examples to facilitate information retrieval from publications using LLM (RAG technique). Below are some examples of CQs:

Example CQs

- What data formats are used in the deep learning pipeline (e.g, image, audio, video, csv)?
- What are the data augmentation techniques applied in the deep learning pipeline (e.g, Flipping, Rotating, Scaling)?
- Which frameworks are used to build the deep learning model (e.g., TensorFlow, PyTorch)?

The complete list of competency questions, along with the purpose, scope, target users, and intended uses of the ontology, can be found in the Ontology Requirement Specification Document (ORSD) (Table 6). The competency questions are grouped into five categories: Data Acquisition, Data Preprocessing, Model Development, Model Evaluation, and Model Usage and Deployment.

3.3. Ontology creation

Following the development of CQs, we conducted experiments for the next steps of our pipeline using four different LLMs: Mixtral 8x22B Instruct v0.1, GPT-4o, GPT-3.5 and Gemini. These experiments aimed to construct the ontology and retrieve KG individuals from the CQ answers using the same prompt for all models.

A two-step strategy was implemented to create the ontology from the 28 available CQs. In the first step, we aimed to extract all concepts, relationships and their properties from the CQs. To achieve this, we included a set of instructions and an example CQ with expected output in the prompt[9]. In the second step, we constructed an ontology for describing information on DL pipelines. This was achieved by providing an in-context example in a prompt[10] containing a basic ontology structure, utilizing the PROV-O ontology [30] as a foundational ontology for reuse, and incorporating the concepts, relationships and their properties extracted from the CQs.

3.4. CQ Answering

This component is the central pillar of the whole pipeline, which connects the retrieved information from the publications to KG creation. We utilized the RAG approach to extract answers to all CQs from the first 30 publications employing DL methods within our 61 biodiversity-focused publication dataset.

We then applied basic text processing steps to refine the generated answers. These steps involved removing redundant and repetitive content where applicable and ensuring that the generated content was informative, noise-free, and ready for integration into KGs. We used only the Mixtral 8x22B Instruct v0.1 model to generate CQ answers, as other selected models are not open-source to implement this

[8] https://github.com/fusion-jena/automatic-KG-creation-with-LLM/blob/master/CQs/CQs.txt

[9] https://github.com/fusion-jena/automatic-KG-creation-with-LLM/tree/master/Ontology/Mixtral_8_22b/Prompt/Concepts_relationships_dataproperties_extraction.txt

[10] https://github.com/fusion-jena/automatic-KG-creation-with-LLM/tree/master/Ontology/Mixtral_8_22b/Prompt/Ontology_creation_with_data_props.txt

process programmatically. Below are random examples of the RAG-generated CQ answers after basic text processing:

Example CQ answers

Question: What are the datasets used in the deep learning pipeline (e.g, MNIST, CIFAR, ImageNet)?
Answer from random publication: The deep learning pipeline uses the CitySounds2017 dataset, an expert-annotated dataset of urban sounds collected across Greater London, UK. This dataset was used to train the CNNs, CityBioNet and CityAnthroNet, for measuring audible (0-12 kHz) biotic and anthropogenic acoustic activity in audio recordings from urban environments.
Answer from random publication: The datasets used in the deep learning pipeline are not explicitly mentioned in the provided context. However, it is mentioned that a time-series of MODIS-driven NDVI and LAI were used in the deep learning-based species richness modeling.
Answer from random publication:The context does not provide information on the specific datasets used in the deep learning pipeline.

3.5. KG construction

CQs selection: It is unlikely for publications to mention details such as deployment status, model randomness, generalizability, and similar aspects. Therefore, for the KG creation step, we select 13 CQs (highlighted CQs in Table 6) out of 28 that we believe have information mostly available in every publication and are high likely to be addressed in typical DL research papers [25]. This selective approach ensures that the KG can be consistently populated with data. The chosen questions cover essential aspects such as data preprocessing, model architecture, training procedures, and evaluation metrics.
Named Entity Recognition (NER): The NER process is conducted on the pre-selected answers to the selected 13 CQs (highlighted CQs in table 6) using a consistent prompt[11] that includes in-context examples with expected outputs. This process is performed across four different LLMs: Mixtral 8x22B Instruct v0.1, GPT-3.5, GPT-4o, and Gemini on Mixtral generated CQ answers. By using the same prompt across all LLMs, we ensure a fair comparison of the results, allowing us to accurately assess the performance and effectiveness of each model in generating relevant and consistent responses.
Mapping entities: After identifying the entities through NER, we map them to the ontology generated by the LLM, described in Section 3.3. This mapping process involves aligning the extracted entities with predefined concepts and relationships within the ontology to ensure accurate representation. We utilized the rdflib[12] Python library for this entity mapping.

3.6. Evaluation (Ontology, RAG generated CQ answers and KG)

Three key outputs produced by the LLMs were evaluated in this step: ontology, generated CQ answers, and the KG individuals, which were automatically extracted from CQ answers. Each ontology generated by the four language models was evaluated using the Ontology Pitfall Scanner (OOPS!) tool [31]. OOPS! is a web-based platform designed to automatically identify potential errors or inconsistencies within ontologies. The tool conducts a comprehensive series of checks and generates a detailed report outlining detected pitfalls, including their specific identification number and a corresponding description.

To evaluate RAG-generated CQ answers, we enlisted a human expert to manually review each publication for the selected 13 CQs used in KG construction. The expert assigned single-word evaluations (True, False, Partly, Generalized) to the RAG-generated CQ answers. Given the time and effort required for manual annotation, the expert annotated 30 scholarly articles. These 30 publications are subsequently

[11]https://github.com/fusion-jena/automatic-KG-creation-with-LLM/blob/master/NER_prompt/NER_Mixtral_prompt.txt
[12]https://pypi.org/project/rdflib/

used in the complete pipeline to test the feasibility of our approach. We believe this evaluation is adequate to establish proof of concept and can be extended to other publications.

To evaluate the knowledge graph (KG), we assessed the accuracy of the KG entities by verifying their presence in the CQ answers using SPARQL queries. This approach allows us to determine whether the extracted and mapped entities accurately reflect the information provided in the original publications. Additionally, we compared the number of entities extracted by each LLM from the 13 CQ answers and calculated the percentage of entity and instance intersection when comparing the results from pairs of the four models. This comparison provides insights into the consistency and reliability of the different LLMs in extracting relevant entities for the KG.

4. Results

This section outlines the outcomes of each pipeline stage. We introduce the DLProv ecosystem, encompassing the selected scholarly publication dataset, competency question answers, ontology, knowledge graph, and evaluation findings. The CQ generation step of the pipeline resulted in 28 CQs, which is outlined in the DLProv Ontology Requirement Specification Document (ORSD) (6). These CQs investigate every step of the DL pipeline, including raw data sources, preprocessing techniques, model architectures, hyperparameter settings, software and hardware choices, post-processing steps, security measures in handling sensitive data, and data biases alongside ethical considerations.

4.1. The DLProv Ontology

Building upon the defined competency questions, the ontology generation step of the pipeline constructed the DLProv Ontology by reusing the PROV-O ontology [30]. Four distinct ontologies were generated using the models: Mixtral 8x22B Instruct v0.1, GPT-4o, GPT-3.5 and Gemini. Figure 2 shows an excerpt of the ontology created using Mixtral 8x22B Instruct v0.1 model.

We then compared the classes, subclasses, object properties, data properties, and axioms generated by each model (Table 1). Among the models employed for ontology generation, Mixtral 8x22B Instruct v0.1 exhibited the most expansive output. It generated a higher number of classes (30) compared to the other models. Similarly, Mixtral 8x22B Instruct v0.1 produced a greater quantity of properties, encompassing both object properties, data properties, and axioms. Following Mixtral 8x22B Instruct v0.1 in terms of output volume were GPT-4o, GPT-3.5, and lastly, Gemini.

Ontology Evaluation: A comparative analysis of the generated ontologies using the OOPS! tool revealed distinct patterns (Table 2). Mixtral 8x22B Instruct v0.1 and GPT-4o exhibited similar pitfall profiles, differing from those identified in GPT-3.5 and Gemini. Notably, GPT-3.5 and Gemini contained a higher frequency of pitfalls overall. A common issue across all models was the inclusion of an isolated ontology element, *prov:Entity*, which unconnects other ontology concepts. While Mixtral 8x22B Instruct v0.1 and GPT-4o predominantly structured their ontologies with all classes as subclasses of *prov:Entity*, both models also exhibited isolated instances of untyped *prov:Entity* classes, based on OOPS! evaluation. None of the generated ontologies included the owl:Ontology declaration for metadata and license, a limitation likely attributed to the absence of specific prompting in this regard.

Model	Classes	Object Properties	Data Properties	SubClassOf	Axiom
Mixtral 8x22B Instruct v0.1	**30**	**58**	**28**	**30**	**548**
GPT-4o	28	27	2	28	216
GPT-3.5	26	52	1	26	210
Gemini	24	46	2	24	189

Table 1
Statistics of the Ontology created by four different state-of-the-art LLMs

Our analysis considers both the statistics and OOPS! evaluation revealed that the ontology generated by Mixtral 8x22B Instruct v0.1 exhibited a higher degree of complexity compared to the other models.

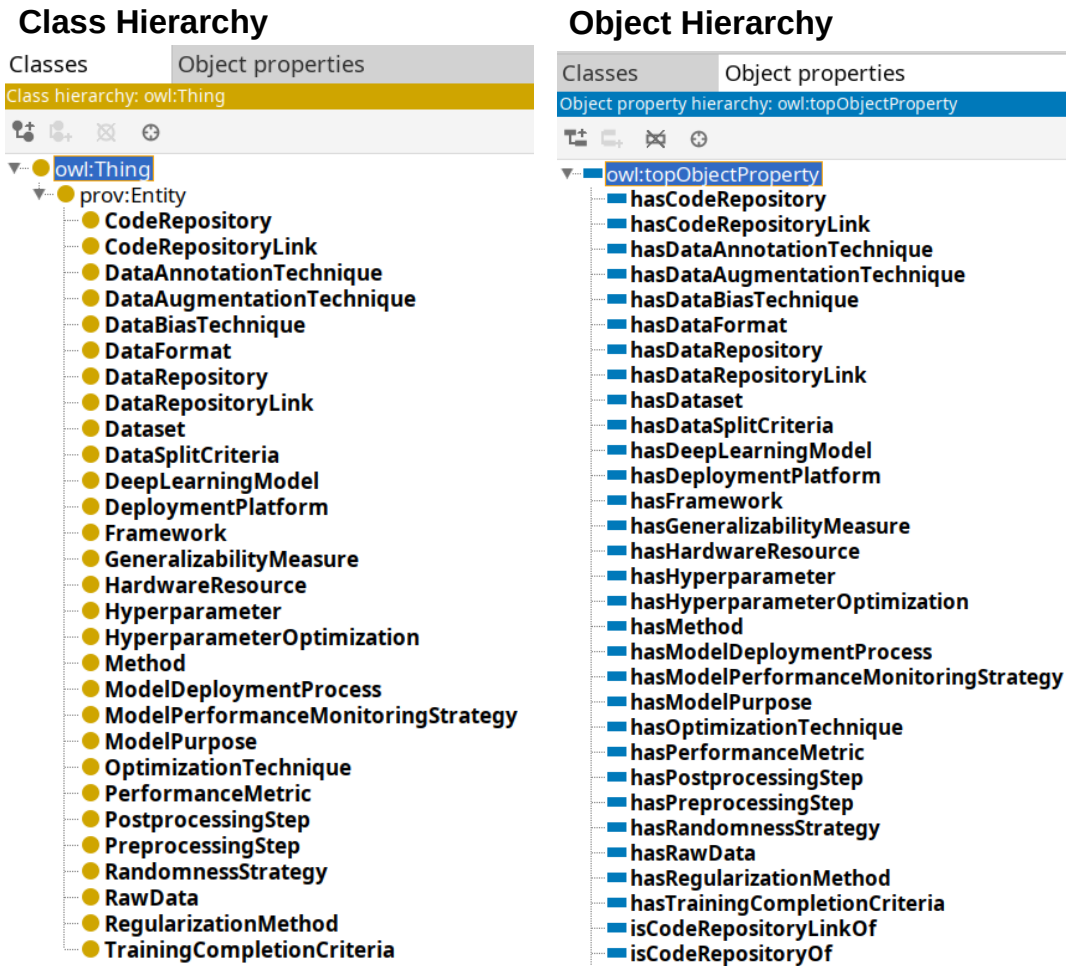

Figure 2: An excerpt of the DLProv ontology generated by the Mixtral 8x22B Instruct v0.1 in our pipeline.

This complexity was evident in the number of classes, subclasses, axioms, and a favourable evaluation by the Ontology Pitfall Scanner. Hence, we use this ontology for named entity mapping in the KG creation step.

While GPT-4o produced comparable results, the cost associated with its OpenAI API limited its feasibility for large-scale processing. Given this constraint, we opted for Mixtral 8x22B Instruct v0.1, an open-source model, for both CQ answer generation from the 30 publications and the subsequent ontology generation used in the KG construction process.

4.2. The DLProv Question Answers

The CQ answering step of the pipeline generated answers to all CQs for the first 30 publications from the selected dataset. We selected only 13 CQs (highlighted CQs in table 6), assumed to be consistently present across publications, for the next step of KG construction. A human expert manually assessed the RAG-generated answers generated by the Mixtral 8x22B Instruct v0.1 model for these 13 CQs across the 30 publications, resulting in 390 evaluations. Answers were categorized into four classes: True (exact match), False (incorrect), Partly (partially correct), and Generalized (correct but lacking specificity or details in the publication). Of the 390 evaluated answers, there were 323 agreements between the human annotator and the LLM retrieved answers, with six instances categorized as partially correct and 11 as generalized.

Error Code	Error Description	Mixtral 8x22B Instruct v0.1	GPT-4o	GPT-3.5	Gemini
P04	Creating unconnected ontology elements			✓	✓
P08	Missing annotations	✓	✓	✓	✓
P10	Missing Disjointness	✓	✓	✓	✓
P11	Missing domain or range in properties			✓	✓
P13	Inverse relationships not explicitly declared	✓	✓	✓	✓
P34	Untype Class	✓	✓		
P38	No OWL Ontology Declaration	✓	✓	✓	✓
P41	No License declared	✓	✓	✓	✓

Table 2
Evaluation of the DLProv Ontology generated by different LLMs with Ontology Pitfall Scanner

4.3. The DLProv KG

Named entities extracted from the selected CQ answers were mapped to ontology concepts derived from the DL pipeline methodologies outlined in 30 scholarly publications. This process resulted in the creation of a distinct DLProv KG for each publication using four different language models, totalling 120 KGs. Listing 1 presents a sample KG illustrating the provenance of DL method results from publication [32].

Listing 1: An excerpt of the KG generated by our (semi-) automated pipeline by Mixtral 8x22B Instruct v0.1 model

```
dlprov:Accuracy a dlprov:PerformanceMetric .
dlprov:Adam a dlprov:OptimizationTechnique .
dlprov:Brightness_adjustments a dlprov:PreprocessingStep .
dlprov:CNN a dlprov:DeepLearningModel .
dlprov:DeepLabv3%2B a dlprov:Method .
```

4.4. KG comparison from different models

With 120 KGs generated — four for each publication using four different language models — we conducted a comparative analysis. This involved assessing the accuracy of entities within each publication's four KGs. Additionally, we examined the intersection and divergence of entities and instances across all possible model combinations. This comparative analysis provides insights into the consistency and variability in entity extraction and representation among different language models.

Table 3 presents a comprehensive overview of concept and instance counts for selected language models across individual publications and the dataset as a whole. The table details the number of concepts and instances extracted by each model for each publication. Notably, Mixtral 8x22B Instruct v0.1 retrieved the maximum number of concepts, identifying a total of 359 concepts across all 30 publications. In contrast, Gemini retrieved the minimum number of concepts, totalling 202. In terms of instances, GPT-4o emerged as the top performer with 720 instances, while Gemini recorded the fewest instances at 445.

Table 4 comprises the total number of KG instances and their percentage that aligns with the RAG-generated CQ answers for each publication. GPT-4 had the highest number of KG instances that are in line with the CQ answers, with 586 out of 720 instances, resulting in an accuracy of 81.39%. Mixtral 8x22B Instruct v0.1 trailed with 378 out of 575 instances, resulting in an accuracy of 65.74%.

Table 5 represents the intersection of concepts and instances between two models for all possible model combinations. The combination of GPT-4 and GPT-3.5 stood at the forefront for concepts and instances, with 75.80% and 32.75% intersections, respectively. At the lowest, for concepts, the Mixtral

8x22B Instruct v0.1 and Gemini model pair stood at 46.09%, and for instances, the GPT-3.5 and Gemini model pair stood at 19.14%.

Table 3: The number of concepts and instances within the knowledge graphs generated by each of the four LLMs for individual publications, along with the total concept count across all publications. The metadata, title and doi of the publications are available on GitHub.

Publication	Mixtral 8x22B Instruct v0.1 Concepts	GPT-4o Concepts	GPT-3.5 Concepts	Gemini Concepts	Mixtral 8x22B Instruct v0.1 Instances	GPT-4o Instances	GPT-3.5 Instances	Gemini Instances
1	8	4	4	4	13	14	9	10
3	8	7	6	5	9	13	9	8
5	9	7	7	6	12	16	13	12
7	11	6	5	3	12	12	9	5
8	20	10	10	7	26	28	24	14
9	14	11	11	10	21	20	23	18
10	8	8	8	6	11	14	12	9
12	11	10	8	9	22	21	20	23
13	7	8	8	4	16	19	20	14
14	8	8	10	5	11	15	20	5
16	10	10	11	8	18	21	22	17
18	13	7	10	4	24	16	14	8
19	6	13	8	5	32	34	18	17
20	28	14	11	7	30	34	30	22
24	7	7	6	5	12	23	18	9
25	28	6	6	2	12	13	13	5
27	15	13	13	3	41	53	46	3
28	5	7	6	3	9	14	11	6
33	8	7	7	4	15	19	13	8
34	7	10	9	8	19	32	28	19
37	15	11	10	9	26	27	26	16
38	8	8	9	9	14	19	19	14
39	24	11	11	10	24	53	39	22
41	13	12	12	11	23	37	25	21
42	15	10	12	10	22	39	38	24
43	10	9	12	11	17	17	23	22
44	14	11	13	10	34	40	41	35
45	9	6	14	8	15	16	22	21
46	9	7	14	8	16	17	20	21
47	11	9	14	8	19	24	13	17
Total	359	267	285	202	575	720	638	445

Table 4: The number of instances in KG that match the RAG-generated CQ answers by four different LLMs. The metadata, title and doi of the publications are available on GitHub.

Publication	Mixtral 8x22B Instruct v0.1	GPT-4o	GPT-3.5	Gemini
1	4/13 (30.77%)	9/14 (64.29%)	7/9 (77.78%)	8/10 (80.00%)
3	4/9 (44.44%)	12/13 (92.31%)	9/9 (100.00%)	8/8 (100.00%)

Continued on next page

Table 4 – Continued from previous page

Publication	Mixtral 8x22B Instruct v0.1	GPT-4o	GPT-3.5	Gemini
5	11/12 (91.67%)	12/16 (75.00%)	11/13 (84.62%)	11/12 (91.67%)
7	4/12 (33.33%)	12/12 (100.00%)	9/9 (100.00%)	5/5 (100.00%)
8	11/26 (42.31%)	25/28 (89.29%)	19/24 1(79.17%)	10/14 (71.43%)
9	16/21 (76.19%)	13/20 1(65.00%)	15/23 (65.22%)	14/18 (77.78%)
10	11/11 (100.00%)	14/14 (100.00%)	10/12 (83.33%)	9/9 1(100.00%)
12	18/22 (81.82%)	18/21 (85.71%)	18/20 1(90.00%)	17/23 (73.91%)
13	16/16 (100.00%)	19/19 1(100.00%)	19/20 (95.00%)	12/14 (85.71%)
14	9/11 1(81.82%)	15/15 (100.00%)	14/20 (70.00%)	4/5 (80.00%)
16	17/18 (94.44%)	15/21 (71.43%)	11/22 (50.00%)	16/17 (94.12%)
18	22/24 (91.67%)	16/16 (100.00%)	11/14 (78.57%)	8/8 (100.00%)
19	29/32 (90.62%)	26/34 (76.47%)	14/18 (77.78%)	16/17 (94.12%)
20	10/30 (33.33%)	27/34 (79.41%)	16/30 (53.33%)	19/22 (86.36%)
24	12/12 (100.00%)	17/23 (73.91%)	14/18 (77.78%)	8/9 (88.89%)
25	3/12 (25.00%)	10/13 (76.92%)	8/13 (61.54%)	5/5 (100.00%)
27	19/41 (46.34%)	47/53 (88.68%)	42/46 (91.30%)	2/3 (66.67%)
28	9/9 (100.00%)	14/14 (100.00%)	10/11 (90.91%)	6/6 (100.00%)
33	6/15 (40.00%)	14/19 (73.68%)	7/13 (53.85%)	5/8 (62.50%)
34	14/19 (73.68%)	19/32 (59.38%)	21/28 (75.00%)	12/19 (63.16%)
37	6/26 (23.08%)	22/27 (81.48%)	18/26 (69.23%)	6/16 (37.50%)
38	12/14 (85.71%)	17/19 (89.47%)	13/19 (68.42%)	5/14 (35.71%)
39	21/24 (87.50%)	38/53 (71.70%)	20/39 (51.28%)	9/22 (40.91%)
41	17/23 (73.91%)	31/37 (83.78%)	16/25 (64.00%)	12/21 (57.14%)
42	14/22 (63.64%)	24/39 (61.54%)	22/38 (57.89%)	14/24 (58.33%)
43	13/17 (76.47%)	14/17 (82.35%)	19/23 (82.61%)	15/22 (68.18%)
44	22/34 (64.71%)	36/40 (90.00%)	35/41 (85.37%)	17/35 (48.57%)
45	3/15 (20.00%)	14/16 (87.50%)	18/22 (81.82%)	6/21 (28.57%)
46	11/16 (68.75%)	15/17 (88.24%)	15/20 (75.00%)	10/21 (47.62%)
47	14/19 (73.68%)	21/24 (87.50%)	8/13 (61.54%)	9/17 (52.94%)
Total	378/575 (65.74%)	586/720 (81.39%)	469/638 (73.51%)	298/445 (66.97%)

Table 5: Intersection of concepts and instances between pairs of models across all possible combinations

Model Combinations	Total number of intersections for all publications	Average number of intersections for all publications
Mixtral 8x22B Instruct v0.1-GPT-4o_Concepts	233/393 (59.29%)	7.77/13.10 (59.29%)
Mixtral 8x22B Instruct v0.1-GPT-3.5_Concepts	224/420 (53.33%)	7.47/14.00 (53.33%)
Mixtral 8x22B Instruct v0.1-Gemini_Concepts	177/384 (46.09%)	5.90/12.80 (46.09%)
GPT-4o-GPT-3.5_Concepts	238/314 (75.80%)	7.93/10.47 (75.80%)

Continued on next page

Table 5 – Continued from previous page

Model Combinations	Total number of intersections for all publications	Average number of intersections for all publications
GPT-4o-Gemini_Concepts	187/282 (66.31%)	6.23/9.40 (66.31%)
GPT-3.5-Gemini_Concepts	186/301 (61.79%)	6.20/10.03 (61.79%)
Mixtral 8x22B Instruct v0.1-GPT-4o_Instances	252/1043 (24.16%)	8.40/34.77 (24.16%)
Mixtral 8x22B Instruct v0.1-GPT-3.5_Instances	229/984 (23.27%)	7.63/32.80 (23.27%)
Mixtral 8x22B Instruct v0.1-Gemini_Instances	185/835 (22.16%)	6.17/27.83 (22.16%)
GPT-4o-GPT-3.5_Instances	335/1023 (32.75%)	11.17/34.10 (32.75%)
GPT-4o-Gemini_Instances	214/951 (22.50%)	7.13/31.70 (22.50%)
GPT-3.5-Gemini_Instances	174/909 (19.14%)	5.80/30.30 (19.14%)

5. Discussion

Despite well-known problems such as hallucination, lack of critical thinking, outdated retrieval, and prompt sensitiveness [33, 3, 34], LLMs are increasingly being used for information retrieval. Initially, we tried to create a KG representing the method information of the DL pipeline from scholarly publications in a single step by providing the whole publication content by prompting the LLM. However, this approach offered little to no content related to KG. Then, we adapted six components of the pipeline: Data Collection, CQ generation, Ontology creation, CQ answering, KG construction, and Evaluation by [7], as discussed in Section 3.

In the entire pipeline, ensuring consistency in the content generated by all LLMs was crucial. Each time we provided input to an LLM, it returned answers in various formats. To address this variability, we incorporated example inputs and expected outputs into the prompts in all possible cases. By doing so, we could enforce consistent formatting in the LLM outputs. However, during the CQ answering step using the RAG technique, we opted not to provide in-context examples to avoid limiting the amount of information retrieved by the LLMs. Consequently, we observed instances of excessive and unnecessary explanation, often starting with certain strings such as "%Explanation Explanation:" or "%Context Context:". We utilized these types of indicators to refine the CQ answers, eliminating redundant and repetitive segments from the generated content. Despite prompting the LLMs to leverage the PROV-O ontology during ontology generation, the resulting ontologies primarily relied on the *prov:Entity* class for subclass creation, exhibiting limited utilization of the broader PROV-O framework.

Evaluating the CQ answers is crucial as they are the connection block between the information within the publication and the KG instances. We investigated different tools like Ragas[13] and Tonic ai[14] to evaluate the RAG-generated content; however, they function fully only when an OpenAI API key is provided. We also tested LLM Judge, where the LLM rates the similarity between the RAG-generated content and the ground truth information [7]. However, we decided to evaluate the RAG-generated content for selected 13 CQs for KG generation with the help of the domain expert by manually going through each publication and then judging the RAG-generated content by annotations.

In our previous work, we provided all question-answer pairs (CQs) and the ontology directly to the model as a prompt for generating KGs using LLMs[7]. However, this approach often resulted in inconsistencies, and LLMs occasionally produced hallucinatory outputs, leading to failures in KG generation in some cases of the results. Now, we have refined the previous approach by extracting named entities from the CQ answers and subsequently mapping them to the ontology. This new method is notably more straightforward and consistently achieves successful KG generation.

As mentioned in the results section 4, we constructed KGs using four different LLMs and compared their concepts and instances in three ways. Our analysis revealed notable variations in the capabilities of the LLMs.

[13] https://docs.ragas.io/en/latest/index.html
[14] https://www.tonic.ai/validate

Specifically, we observed that Mixtral 8x22B Instruct v0.1 and GPT-4o exhibited superior performance in terms of the number of concepts and instances they could generate.

When evaluating the accuracy of the KG instances aligned with the CQs, GPT-4o demonstrated superior performance. Additionally, in terms of intersected concepts and instances between models, the GPT-4o and GPT-3.5 pair showed the maximum intersection. These findings suggest that GPT-4o, in particular, is highly effective in capturing and representing complex entities within Knowledge Graphs. This indicates its potential for enhanced consistency and reliability in KG generation compared to other models. However, the open-source Mixtral 8x22B Instruct v0.1 model's strong performance in concept retrieval and availability for the community highlights its importance and utility, where open, adaptable AI tools are preferable.

6. Conclusion

In this study, we have explored using four different state-of-the-art LLMs: Mixtral 8x22B Instruct v0.1, GPT-3.5, GPT-4o, and Gemini to create ontologies and knowledge graphs. Our proposed pipeline follows established ontology engineering practices and shows the potential of LLMs acting as assistants (or co-pilots) to human experts. With this, ontology and KG creation require significantly lower effort and less semantic web expertise, making these powerful tools available for broader use.

Our findings identified the variability in the generated content from different LLMs at each step of the pipeline. Specifically, Mixtral 8x22B Instruct v0.1 excelled in ontology creation, generating the maximum number of classes, properties, and axioms, and producing consistent formats with fewer errors. It also performed well in concept retrieval. On the other hand, GPT-4o was superior in retrieving KG instances from text data. The other two LLMs followed in performance. This suggests that there is potential in using different LLMs at various stages to maximize the efficiency and quality of ontology and KG creation.

In our future work, we aim to leverage these findings in our ontology and KG development efforts. Additionally, we will explore methods for integrating the generated ontologies with other machine learning (ML) and deep learning (DL) ontologies to enhance interoperability and applicability in broader ML/DL contexts. Overall, our findings highlight the promise of LLMs in assisting with ontology and KG creation, offering a streamlined and efficient approach that can benefit a wide range of applications and users.

Acknowledgments

Supported by the German Centre for Integrative Biodiversity Research (iDiv) Halle-Jena-Leipzig, which is funded by the German Research Foundation (DFG) under FZT 118 (ID 202548816) and also supported by the DFG project TRR 386 (ID 514664767). The authors gratefully acknowledge the computing time granted by the Resource Allocation Board and provided on the supercomputer Emmy/Grete at NHR@Göttingen as part of the NHR infrastructure. The calculations for this research were in part conducted with computing resources under the project nhr_th_starter_22233. In addition to NHR, Friedrich Schiller University Jena also supported with computational infrastructure to create and execute the part of the pipeline in Draco cluster, computer node with GPU accelerator, 2x NVIDIA A100, 80GB.

References

[1] A. Hogan, E. Blomqvist, M. Cochez, C. d'Amato, G. D. Melo, C. Gutierrez, S. Kirrane, J. E. L. Gayo, R. Navigli, S. Neumaier, et al., Knowledge graphs, ACM Computing Surveys (Csur) 54 (2021) 1–37.

[2] M. Funk, S. Hosemann, J. C. Jung, C. Lutz, Towards ontology construction with language models, in: Joint proceedings of the 1st workshop on Knowledge Base Construction from Pre-Trained Language Models (KBC-LM) and the 2nd challenge on Language Models for Knowledge Base Construction (LM-KBC) co-located with the 22nd International Semantic Web Conference (ISWC 2023), Athens, Greece, November 6, 2023, volume 3577 of *CEUR Workshop Proceedings*, CEUR-WS.org, 2023. URL: https://ceur-ws.org/Vol-3577/paper16.pdf.

[3] J. Z. Pan, S. Razniewski, J.-C. Kalo, S. Singhania, J. Chen, S. Dietze, H. Jabeen, J. Omeliyanenko, W. Zhang, M. Lissandrini, R. Biswas, G. de Melo, A. Bonifati, E. Vakaj, M. Dragoni, D. Graux, Large Language Models and Knowledge Graphs: Opportunities and Challenges, Transactions on Graph Data and Knowledge 1 (2023) 2:1–2:38. URL: https://drops.dagstuhl.de/entities/document/10.4230/TGDK.1.1.2. doi:10.4230/TGDK.1.1.2.

[4] S. Haussmann, O. Seneviratne, Y. Chen, Y. Ne'eman, J. Codella, C.-H. Chen, D. L. McGuinness, M. J. Zaki, Foodkg: a semantics-driven knowledge graph for food recommendation, in: The Semantic Web–ISWC 2019: 18th International Semantic Web Conference, Auckland, New Zealand, October 26–30, 2019, Proceedings, Part II 18, Springer, 2019, pp. 146–162.

[5] T. Brown, B. Mann, N. Ryder, M. Subbiah, J. D. Kaplan, P. Dhariwal, A. Neelakantan, P. Shyam, G. Sastry, A. Askell, et al., Language models are few-shot learners, Advances in neural information processing systems 33 (2020) 1877–1901.

[6] A. Chowdhery, S. Narang, J. Devlin, M. Bosma, G. Mishra, A. Roberts, P. Barham, H. W. Chung, C. Sutton, S. Gehrmann, et al., Palm: Scaling language modeling with pathways, Journal of Machine Learning Research 24 (2023) 1–113.

[7] V. K. Kommineni, B. König-Ries, S. Samuel, From human experts to machines: An llm supported approach to ontology and knowledge graph construction, 2024. URL: https://arxiv.org/abs/2403.08345. arXiv:2403.08345.

[8] H. Liu, Y. Perl, J. Geller, Concept placement using BERT trained by transforming and summarizing biomedical ontology structure, Journal of Biomedical Informatics 112 (2020) 103607.

[9] F. Petroni, T. Rocktäschel, S. Riedel, P. Lewis, A. Bakhtin, Y. Wu, A. Miller, Language models as knowledge bases?, in: Proceedings of the 2019 Conference on Empirical Methods in Natural Language Processing and the 9th International Joint Conference on Natural Language Processing (EMNLP-IJCNLP), Association for Computational Linguistics, Hong Kong, China, 2019, pp. 2463–2473. URL: https://aclanthology.org/D19-1250. doi:10.18653/v1/D19-1250.

[10] L. Yao, C. Mao, Y. Luo, KG-BERT: BERT for knowledge graph completion, arXiv preprint arXiv:1909.03193 (2019).

[11] L.-P. Meyer, C. Stadler, J. Frey, N. Radtke, K. Junghanns, R. Meissner, G. Dziwis, K. Bulert, M. Martin, LLM-assisted knowledge graph engineering: Experiments with chatgpt, arXiv preprint arXiv:2307.06917 (2023).

[12] Y. Rebboud, L. Tailhardat, P. Lisena, R. Troncy, Can LLMs Generate Competency Questions?, in: ESWC 2024, Extended Semantic Web Conference, Hersonissos, Greece, 2024. URL: https://hal.science/hal-04564055.

[13] R. Cohen, M. Geva, J. Berant, A. Globerson, Crawling the internal knowledge-base of language models, in: Findings of the Association for Computational Linguistics: EACL 2023, Dubrovnik, Croatia, May 2-6, 2023, Association for Computational Linguistics, 2023, pp. 1811–1824. URL: https://doi.org/10.18653/v1/2023.findings-eacl.139. doi:10.18653/V1/2023.FINDINGS-EACL.139.

[14] A. Zaitoun, T. Sagi, S. Wilk, M. Peleg, Can large language models augment a biomedical ontology with missing concepts and relations?, arXiv preprint arXiv:2311.06858 (2023).

[15] S. Toro, A. V. Anagnostopoulos, S. Bello, K. Blumberg, R. Cameron, L. Carmody, A. D. Diehl, D. Dooley, W. Duncan, P. Fey, et al., Dynamic retrieval augmented generation of ontologies using artificial intelligence (dragon-ai), arXiv preprint arXiv:2312.10904 (2023).

[16] H. Babaei Giglou, J. D'Souza, S. Auer, LLMs4OL: Large language models for ontology learning, in: International Semantic Web Conference, Springer, 2023, pp. 408–427.

[17] B. Veseli, S. Singhania, S. Razniewski, G. Weikum, Evaluating language models for knowledge base completion, in: The Semantic Web - 20th International Conference, ESWC 2023, Hersonissos, Crete, Greece, May 28 - June 1, 2023, Proceedings, volume 13870 of *Lecture Notes in Computer Science*, Springer, 2023, pp. 227–243. URL: https://doi.org/10.1007/978-3-031-33455-9_14. doi:10.1007/978-3-031-33455-9_14.

[18] F. Neuhaus, Ontologies in the era of large language models - a perspective, Appl. Ontology 18 (2023) 399–407. URL: https://doi.org/10.3233/AO-230072. doi:10.3233/AO-230072.

[19] G. C. Publio, D. Esteves, A. Ławrynowicz, P. Panov, L. Soldatova, T. Soru, J. Vanschoren, H. Zafar, Ml-schema:

exposing the semantics of machine learning with schemas and ontologies, arXiv preprint arXiv:1807.05351 (2018).
[20] C. M. Keet, A. Ławrynowicz, C. d'Amato, A. Kalousis, P. Nguyen, R. Palma, R. Stevens, M. Hilario, The data mining optimization ontology, Journal of web semantics 32 (2015) 43–53.
[21] P. Panov, L. Soldatova, S. Džeroski, Ontology of core data mining entities, Data Mining and Knowledge Discovery 28 (2014) 1222–1265.
[22] D. Esteves, D. Moussallem, C. B. Neto, T. Soru, R. Usbeck, M. Ackermann, J. Lehmann, Mex vocabulary: a lightweight interchange format for machine learning experiments, in: Proceedings of the 11th International Conference on Semantic Systems, 2015, pp. 169–176.
[23] I. Dasoulas, D. Yang, A. Dimou, Mlsea: A semantic layer for discoverable machine learning, in: European Semantic Web Conference, Springer, 2024, pp. 178–198.
[24] W. Ahmed, V. K. Kommineni, B. König-ries, S. Samuel, How reproducible are the results gained with the help of deep learning methods in biodiversity research?, Biodiversity Information Science and Standards 7 (2023).
[25] W. Ahmed, V. K. Kommineni, B. König-Ries, J. Gaikwad, L. M. R. G. Jr., S. Samuel, Evaluating the method reproducibility of deep learning models in the biodiversity domain, CoRR abs/2407.07550 (2024). URL: https://doi.org/10.48550/arXiv.2407.07550. doi:10.48550/ARXIV.2407.07550. arXiv:2407.07550.
[26] N. Abdelmageed, F. Löffler, L. Feddoul, A. Algergawy, S. Samuel, J. Gaikwad, A. Kazem, B. König-Ries, BiodivNERE: Gold standard corpora for named entity recognition and relation extraction in the biodiversity domain, Biodiversity Data Journal 10 (2022).
[27] O. E. Gundersen, S. Shamsaliei, R. J. Isdahl, Do machine learning platforms provide out-of-the-box reproducibility?, Future Generation Computer Systems 126 (2022) 34–47.
[28] J. Pineau, P. Vincent-Lamarre, K. Sinha, V. Larivière, A. Beygelzimer, F. d'Alché Buc, E. Fox, H. Larochelle, Improving reproducibility in machine learning research (a report from the neurips 2019 reproducibility program), The Journal of Machine Learning Research 22 (2021) 7459–7478.
[29] S. Samuel, F. Löffler, B. König-Ries, Machine learning pipelines: Provenance, reproducibility and FAIR data principles, in: B. Glavic, V. Braganholo, D. Koop (Eds.), Provenance and Annotation of Data and Processes - 8th and 9th International Provenance and Annotation Workshop, IPAW 2020 + IPAW 2021, Virtual Event, July 19-22, 2021, Proceedings, volume 12839 of *Lecture Notes in Computer Science*, Springer, 2021, pp. 226–230. URL: https://doi.org/10.1007/978-3-030-80960-7_17. doi:10.1007/978-3-030-80960-7_17.
[30] T. Lebo, S. Sahoo, D. McGuinness, K. Belhajjame, J. Cheney, D. Corsar, D. Garijo, S. Soiland-Reyes, S. Zednik, J. Zhao, PROV-O: The PROV ontology, W3C recommendation 30 (2013).
[31] M. Poveda-Villalón, A. Gómez-Pérez, M. C. Suárez-Figueroa, OOPS! (OntOlogy Pitfall Scanner!): An On-line Tool for Ontology Evaluation, International Journal on Semantic Web and Information Systems (IJSWIS) 10 (2014) 7–34.
[32] B. R. Hussein, O. A. Malik, W.-H. Ong, J. W. F. Slik, Automated extraction of phenotypic leaf traits of individual intact herbarium leaves from herbarium specimen images using deep learning based semantic segmentation, Sensors 21 (2021). URL: https://www.mdpi.com/1424-8220/21/13/4549. doi:10.3390/s21134549.
[33] Y. Lu, M. Bartolo, A. Moore, S. Riedel, P. Stenetorp, Fantastically ordered prompts and where to find them: Overcoming few-shot prompt order sensitivity, in: Proceedings of the 60th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers), Association for Computational Linguistics, Dublin, Ireland, 2022, pp. 8086–8098. URL: https://aclanthology.org/2022.acl-long.556. doi:10.18653/v1/2022.acl-long.556.
[34] S. Razniewski, A. Yates, N. Kassner, G. Weikum, Language models as or for knowledge bases, arXiv preprint arXiv:2110.04888 (2021).

Table 6: The DLProv Ontology Requirements Specification Document

The DLProv Ontology Requirements Specification Document
1. Purpose
The purpose of this ontology is to provide a knowledge model for the Deep Learning (DL) domain. The ontology aims to facilitate understanding, knowledge sharing, interoperability, and integration of resources related to DL.
2. Scope
The ontology has to focus on the concepts, relationships, algorithms, models, tools, and datasets for DL.
3. Implementation Language
The ontology will be implemented in OWL language.
4. Intended End-Users
User 1. Researchers and academics conducting DL reproducibility studies. User 2. Developers and Practitioners to aid in the development, deployment, and maintenance of DL models. User 3. Organizations for integrating Deep Learning capabilities into their systems and promoting standardization.
5. Intended Uses
Use 1. Facilitating reproducibility of DL studies by providing a standardized framework for organizing knowledge. Use 2. Enhancing searchability and retrieval of DL resources. Use 3. Facilitating interoperability among different systems and tools used in Deep Learning. Use 4. Assisting practitioners in selecting appropriate models, algorithms, and tools based on specific requirements.
6. Ontology Requirements
a. Non-Functional Requirements
NFR 1. The ontology must be written in English. NFR 2. The ontology must reuse other ontologies if required.
b. Functional Requirements: Groups of Competency Questions
CQG1. Data Acquisition
CQ1. What methods are utilized for collecting raw data in the deep learning pipeline (e.g., surveys, sensors, public datasets)? **CQ2. What are the datasets used in the deep learning pipeline (e.g, MNIST, CIFAR, ImageNet)?** CQ3. Where is the data repository of the deep learning pipeline available (e.g, Zenodo, Figshare, Dryad, GBIF)? CQ4. What is the data repository link of the deep learning pipeline (e.g, Link to Zenodo, Figshare, Dryad, GBIF)?
CQG2. Data Preprocessing
CQ5. What data formats are used in the deep learning pipeline (e.g, image, audio, video, csv)? CQ6. What are the data annotation techniques used in the deep learning pipeline (e.g, bounding box annotation, instance segmentation)? **CQ7. What are the data augmentation techniques applied in the deep learning pipeline (e.g, Flipping, Rotating, Scaling)?** **CQ8. What preprocessing steps are involved before training a deep learning model (e.g., normalization, scaling, cleaning)?** CQ9. What is the criteria used to split the data for deep learning model training (e.g., train, test, validation)? CQ10. What techniques are used to address data bias during preprocessing of deep learning pipeline (e.g., Stratified splitting, oversampling, undersampling, Diverse data collection)?
CQG3. Model Development
CQ11. What type of deep learning model is used in the pipeline (e.g., CNN, RNN, Transformer)? **CQ12. What are the hyperparameters used in the deep learning model (e.g., learning rate, optimizer)**? CQ13. How are the hyperparameters of the model optimized (e.g., grid search, random search)? **CQ14. What optimization techniques are applied in the deep learning pipeline (e.g., SGD, Adam)?** CQ15. What criteria are used to determine when training is complete (e.g., validation loss plateau)? **CQ16. What are the regularization methods used to prevent overfitting in the deep learning pipeline (e.g., dropout, L2 regularization)?** CQ17. What is the strategy implemented to monitor the model performance during training? **CQ18. Which frameworks are used to build the deep learning model (e.g., TensorFlow, PyTorch)?** **CQ19. Which hardware resources are used for training the deep learning model (e.g., GPUs, TPUs)?**
CQG4. Model Evaluation
CQ20. What are the postprocessing steps involved after the model training (e.g., Saliency maps, Metrics calculation, Confusion matrix)? **CQ21. What metrics are used to evaluate the performance of the deep learning model (e.g., accuracy, precision, recall)?** CQ22. What measures were taken to ensure the generalizability of the deep learning model (e.g., Diverse dataset, Cross Validation, Stratified splitting)? CQ23. What strategies are employed to handle randomness in the deep learning pipeline (e.g., random seed value)?
CQG5. Model Usage and Deployment

CQ24. What is the purpose of the deep learning model (e.g., classification, segmentation, detection)?
CQ25. Where is the code repository of the deep learning pipeline available (e.g, GitHub, GitLab, BitBucket)?
CQ26. What is the code repository link of the deep learning pipeline (e.g, Link to GitHub, GitLab, BitBucket)?
CQ27. What process was followed to deploy the trained deep learning model (e.g., Model serialization, Platform selection)?
CQ28. Which platform was used to deploy the deep learning model (e.g., AWS, Azure, Google cloud platform)?

7. Pre-Glossary of Terms

a. Terms from Competency Questions + Frequency

Deep learning 24	Training 6	Deploy 2	Metric 2
Model 15	Process 3	Preprocessing 2	Label 2
Pipeline 14	Dataset 3	Step 2	Hyperparameter 2
Data 9	Repository 4	Method 2	Code 2

b. Objects

No objects were identified.