

Knowledge Engineering with Large Language Models: A Capability Assessment in Ontology Evaluation

Stefani Tsaneva^{a,*}, Guntur Budi Herwanto^b, Majlinda Llugiqi^a and Marta Sabou^a

^a *Institute of Data, Process and Knowledge Management, Vienna University of Economics and Business, Austria*
E-mails: stefani.tsaenva@wu.ac.at, majlinda.llugiqi.rexha@wu.ac.at, marta.sabou@wu.ac.at

^b *Department of Computer Science and Electronics, Universitas Gadjah Mada, Indonesia*
E-mail: gunturbudi@ugm.ac.id

Abstract. Advancements in large language models (LLMs) offer opportunities for automating challenging and time-intensive Knowledge Engineering (KE) tasks. Constructing an ontology is a complex process, particularly when logical restrictions are modeled or when the development is performed by novice knowledge engineers or domain experts with limited training in KE. Consequently, developed ontologies often contain modeling errors, undermining the success of ontology-based applications and hindering subsequent KE tasks. Thus, it is important to investigate how LLMs can support KE tasks such as the evaluation of ontologies, involving the detection and correction of errors in knowledge-based resources. However, challenges remain in systematically evaluating LLM performance and comparing different models in terms of their capabilities to perform concrete KE tasks. Moreover, there is a lack of comprehensive, task-specific benchmarks needed for such LLM capability assessments. As a result, selecting the right LLM to effectively support knowledge engineers presents a nontrivial problem. To fill these gaps, this study investigates how and to what extent LLMs can support four concrete ontology evaluation sub-tasks: the detection, classification, explanation, and possible correction of modeling issues in ontologies, focusing on the use of existential, universal, and cardinality constraints. To this end, we construct a benchmark dataset based on student-built ontologies and perform experimental assessments of the performance of four LLMs—GPT-4o, Claude Sonnet, DeepSeek V3, and Llama 3.3—on these four KE sub-tasks. Additionally, we exemplify the definition of an annotation framework for the qualitative evaluation of LLM outputs and perform a comparative analysis of each model’s capabilities. Our findings reveal notable differences in model behavior and task-specific strengths, underscoring the importance of selecting the most appropriate model to support concrete KE tasks.

Keywords: knowledge engineering, ontology evaluation, ontology modeling defects, ontology benchmark, large language models, LLM output evaluation

1. Introduction

The rapid progress in Generative AI, particularly through large language models (LLMs), is reshaping the field of Knowledge Engineering (KE) by enabling new research and application opportunities [3, 15, 18]. A typical KE process involves a variety of activities such as knowledge acquisition and representation, for instance in an ontology [25]. As such, it traditionally requires extensive manual efforts for the definition, implementation and validation of domain-specific requirements. However, tool support remains limited for various KE tasks [36], leading to potential modeling mistakes, particularly when curators have limited KE expertise and must handle complex logical

*Corresponding author. E-mail: stefani.tsaenva@wu.ac.at.

constraints [22]. While automated methods have been developed to assist certain tasks, such as checking ontologies for logical errors [20], a logically consistent ontology is not necessarily free from modeling inaccuracies. For instance, the constraint “*Every FamousActor plays a MainCharacter*” could be modeled by defining an equivalence between the class *FamousActor* and an *Actor*, who plays some *MainCharacter*. However, this implementation additionally introduces the unintended restriction that every *Actor* who plays a *MainCharacter* is a *FamousActor*. The detection of such issues typically requires human domain expertise [20]. Yet, a manual validation approach becomes unfeasible for large-scale resources [17, 29], highlighting the need for automated scalable support of this KE task.

Recently, LLMs have been shown to match human performance in many natural language processing tasks minimizing the reliance on human input [5, 24]. Promising results have been reported in the KE field across a range of tasks. An actively explored research task is the use of LLMs in constructing and completing knowledge graphs (KGs) [4, 26, 35, 37]. LLMs have been additionally leveraged to support the evaluation of semantic artifacts by assessing ontology requirements through competency question validation [30] and coverage analyses [36]. KG triple validation has been addressed using LLM-supported tools [2, 12, 29] and LLMs have been incorporated for ontology quality evaluation workflows including the detection of defects [28] and the correction of quality issues identified through external services [7].

Despite the growing interest in leveraging LLMs for KE tasks, several critical gaps remain unaddressed: First (Problem P1), tasks such as ontology evaluation remain underexplored [9]. Existing studies [7, 28] often lack systematic evaluation using datasets from diverse domains, limiting our understanding of how well current methods generalize across different contexts. Second (P2), for ontology evaluation and other KE tasks, it is challenging to determine which LLMs are most suitable. Many studies rely on a single model (e.g., for competency question and user story generation [36]; for ontology generation [7] and for ontology defect detection [28]) without comparing performance across different LLMs. This lack of comparative analysis creates a dependency on specific models, reducing reproducibility and limiting generalisability. Third (P3), the effective assessment of LLMs for KE requires well-designed benchmarks tailored to specific tasks. However, such benchmarks are often unavailable. For instance, in ontology evaluation, there is no comprehensive benchmark incorporating a variety of modeling defects, hampering evaluation and systematic comparison of approaches. Fourth (P4), evaluating LLM outputs presents a significant challenge due to the absence of standardized metrics and methods to assess their performance effectively across KE tasks. While certain tasks, such as binary triple validation, can rely on gold-standard datasets, others—such as ontology generation or the explanation of ontology modeling defects—require expert judgment. This reliance on manual evaluation efforts makes large-scale and consistent assessment of LLM usage unfeasible.

To address these research gaps, this study conducts experimental investigations into the capabilities of LLMs for the task of ontology evaluation. In particular, we are guided by the overarching research question: *How and to what extent can LLMs be employed for ontology evaluation?* We break this down into the following specific research questions:

- *RQ1: Which ontology evaluation sub-tasks can LLMs effectively support?*
- *RQ2: Do certain LLMs exhibit specific strengths or limitations in ontology evaluation tasks?*
- *RQ3: Are there observable behavioral differences across models in how they perform ontology evaluation?*
- *RQ4: How can we define objective assessment criteria to guide expert annotation of LLM-generated outputs?*

To answer these research questions, we scope the problem to ontology evaluation in the context of detecting, explaining, and possibly correcting modeling defects in ontology axioms that involve cardinality, universal, and existential restrictions. Specifically, we focus on defects that arise from the incorrect or inappropriate use of these constraint types. We make the following contributions:

- *Ontology Evaluation Sub-Tasks Definition.* We define four ontology evaluation sub-tasks: (1) detecting when an ontology axiom does not accurately model an intended constraint, (2) classifying a modeling issue according to a predefined set of mistake types, (3) explaining the identified modeling issue, and (4) generating a correct modeling alternative that properly captures the intended constraint.
- *Ontology Benchmark Creation.* To support systematic evaluation of these tasks, we construct a benchmark dataset using student-built ontologies, which contain a diverse range of modeling issues, representing a variety of domains. A subset of this benchmark is manually annotated by experts to enable gold-standard evaluation for the modeling issue detection and classification tasks.

- 1 – *Experimental LLM Capability Assessment.* We conduct four experiments—one for each sub-task—to assess the capabilities of LLMs in isolation. We include four LLMs in our study, two proprietary models (GPT 4o and Claude Sonnet) and two open source models (DeepSeek V3 and Llama 3.3).
- 2 – *Guided Expert-Annotation Schema for LLM Outputs.* To evaluate generated modeling mistake explanations—where comparison to a predefined gold standard is unfeasible due to the diversity of valid outputs, we design an expert annotation schema. The schema guides expert annotations based on key output criteria, facilitating a reproducible and consistent assessment of LLM generations.
- 3 – *Comprehensive and Comparative Evaluation.* Finally, we analyze the results across LLMs to enable a comparative evaluation of model performance and to identify consistent strengths, weaknesses, and behavioral patterns when addressing the four ontology evaluation tasks.

To the best of our knowledge, we pioneer extended experiments for LLM-based ontology evaluation by systematically assessing LLM capabilities across four specific sub-tasks. The comprehensive analysis of our experimental investigation results leads to the following findings:

- 4 – *Varying Performance Across LLMs and Ontology Evaluation Tasks (RQ1;RQ2).* Our findings indicate that some LLMs perform significantly better than others on certain evaluation tasks. For instance, DeepSeek V3 outperforms other models on the task of detecting a modeling issue, reaching a 65.71% F1 score. GPT-4o performs best on modeling issue classification with a 68.03% F1 score. Modeling mistake explanations generated by Claude Sonnet achieve 91.95% of the total points according to our custom assignment criteria introduced in this paper, and the alternative constraint modelings by this LLM result in the highest accuracy of 79.27%.
- 5 – *Diverse Model Behavior Traits and Task Suitability (RQ2;RQ3).* Understanding the behavior traits of each model is crucial for determining which tasks they can effectively support and which specific LLM is best suited for a given task. For instance, *GPT-4o excels in following instructions precisely*, making it particularly effective for issue classification tasks where accuracy and adherence to guidelines are critical. However, tasks such as generating alternative constraint models require a higher degree of flexibility. *Claude Sonnet demonstrates higher flexibility* by adapting instructions to the context and making informed modifications, which proved beneficial in adding constraints to a base ontology that might not include the necessary structure. These insights highlight the importance of selecting the appropriate LLM based on the specific requirements of each ontology evaluation task.
- 6 – *Objective Evaluation of LLM Outputs benefits from an Assessment Framework (RQ4).* Precisely defining key output requirements and assigning scores to each such criteria facilitated an objective annotation of LLM-generated explanations of modeling mistakes. Applying this approach resulted in perfect inter-rater agreement on a 10% subset of annotations, demonstrating the method’s reliability and reproducibility.

The remainder of this paper is structured as follows: Section 2 reviews related work on the application of LLMs in KE and other relevant fields. In Section 3, we introduce four ontology sub-evaluation tasks. We discuss the construction of a benchmark enabling the experimental investigation of these tasks in Section 4 and continue with a detailed description of our experimental setup in Section 5. Section 6 presents the performance results across LLMs, and concrete LLM usage recommendations are discussed in Section 7. Finally, we conclude with closing remarks in Section 8.

2. Related Work

In this section, we analyze existing LLM-based approaches in the KE field with a focus on the evaluation of semantic resources, i.e., ontologies, knowledge graphs, etc. (Section 2.1). Additionally, we discuss different methods applied for the evaluation of LLM generated outputs, both in the KE and other fields (Section 2.2).

2.1. LLMs for Knowledge Engineering

The use of LLMs in KE—particularly for initial tasks such as requirement specification or semantic resource creation—has inspired numerous experimental investigations. However, the evaluation of these resources remains

an underexplored area [9] and has only recently gained research attention. In this section, we detail 1) approaches that, at least partially, address the evaluation of semantic artifacts and 2) works specifically investigating how LLMs can support the definition of logical ontology constraints.

LLM-Based Semantic Resource Evaluation. Despite their importance, semantic resource evaluation studies remain limited. In [2] the authors explore the use of LLMs to detect missing and incorrect class membership relations in public knowledge graphs. However, the scope is constrained by a small dataset and a limited number of investigated relation types. In [29], a binary validation of generated KG triples is performed by various LLMs, human-in-the-loop and a combination of both agent types. The work focuses on the design and comparison of various interaction workflows between the different validation agents. In [23], the validation of triples by open source LLMs has been explored. In particular, the authors approach the LLM-based validation of four problems arising from automated triple generation: misalignment between the generated triples and ontology class and property definitions, duplication of URIs, semantic contradictions, and syntax validity.

Evaluation tasks have been incorporated as secondary steps within broader workflows in several knowledge engineering approaches. For instance, NeonGPT [7], an LLM-based framework following the Ontology Development 101 methodology [16], supports not only ontology creation but also evaluation. The authors integrate an ontology reasoner and the ontology pitiful scanner (OOPS!) [20] to identify errors, while leveraging the LLM for error correction. However, the paper does not provide a quantitative assessment of the LLM's effectiveness in correcting defects. Similarly, in [12], a triple validation step is integrated into a knowledge graph generation workflow. Yet, the paper does not report a quantitative assessment of their validation approach nor compare it against existing validation methods.

LLM-Based Ontology Constraint Generation and Evaluation. There is limited research on the generation and evaluation of logical ontology constraints using LLMs. In [6], the NeonGPT framework [7] is extended to improve the quality and expressivity of domain-specific ontologies through various prompting strategies. Although the extended framework generates logical axioms, the study does not focus on assessing their quality. Instead, the evaluation of generated ontologies relies primarily on ontology metrics, such as the number of logical axioms produced. In [14], the authors present a Protegé plugin, enabled by an LLM, which allows users to define logical axioms in natural language. The plugin supports both simple instance definitions as well as more complex class axioms, including, for instance, cardinality restrictions. The LLM translates these natural language constraint descriptions into Functional OWL syntax, while external libraries facilitate the integration of the constraint into the ontology. The authors do not include a quantitative analysis of the generations, although such an evaluation is planned for future work. The capabilities of LLMs to understand description logic (DL) ontologies have been empirically investigated in [31]. The authors assess the models' abilities to detect syntactic issues in DL-Lite ontologies, determine the correctness of logical deductions, explain performed inferences, and check the satisfiability of logical axioms. Experimental results show that LLMs perform well in syntax checking and inconsistency detection for small ontologies. However, their performance is limited when handling larger and more complex ontologies involving various constructors such as transitivity characteristics, subsumptions assertions and qualifiers. Finally, the use of LLMs to identify ontology modeling defects, caused by incorrect usage of the existential and universal restrictions, is explored in [28]. While the study reports a validation accuracy of 96%, its scope is limited on a small dataset from the Pizza Ontology¹.

While the reviewed literature indicates the promising potential of LLMs for validating semantic resources, most approaches are in the early stages of development. Furthermore, these approaches are typically tested on small datasets [2, 7, 28], focus on a single domain [7, 29], or lack comprehensive experimental evaluation [6, 7, 14]. In this paper, we aim to address these gaps by performing a number of in-depth experimental investigations of LLM capabilities in assessing ontology axioms using a larger dataset covering various domains.

2.2. LLM Evaluation Approaches

While the use of LLMs has been explored for various tasks in the KE field, there is a lack of standardized methodologies for evaluating the outputs generated by LLMs. Without consistent evaluation practices it becomes

¹Pizza Ontology - <https://protege.stanford.edu/ontologies/pizza/pizza.owl>

difficult to compare methods or ensure the quality of generated ontological content. This is especially important for the evaluation of the experiments conducted in our study, as we aim to assess the quality of generated ontology axioms and modeling issue explanations in a systematic manner. In this section, we discuss different approaches for assessing LLM-generated outputs, both within the KE domain and across other fields.

LLM Capability Assessment Methods in Knowledge Engineering. LLM evaluation approaches in the KE field are diverse and vary according to the generative task at hand. To illustrate the diversity of assessment methods we provide an overview of several such methods.

- *Structural Comparison to Original Resources.* One common evaluation method involves the comparison of previously manually produced resources (e.g., ontologies, user stories, etc.) to LLM-generated resources. For instance, in the evaluation of NeonGPT [7] the authors aim to reproduce the Wine Ontology² and provide a structural comparison between the original and LLM-produced ontology. The same evaluation method is applied for the evaluation of ontologies from more complex domains such as heart disease [13] and groundwater ecosystems [6]. However, this approach primarily assesses the similarity to an existing (high-quality) resource, rather than directly capturing the quality of the generative output itself.
- *Expert Assessments.* During the evaluation of OntoChat [36] domain experts and knowledge engineers were asked to assess the usability of the tool and the quality of the resources generated by it through a survey. However, such an evaluation is expensive, time-consuming and can be rather subjective, especially, when no concrete quality dimensions/quality scores are clearly and thoroughly defined.
- *Gold Standard Dataset Utilization.* In some cases, an existing gold standard, previously created by domain experts, was utilized. Such an assessment can be especially useful for the evaluation of controlled generative outputs, such as binary triple validation [29] or ontology defect detection from pre-defined defect taxonomy [28]. However, such an approach is not suitable for tasks such as generating ontology modeling or explaining a detected mistake where a range of possible valid outputs exists.
- *Task benchmarks* have also been created to test the abilities of LLMs in smaller well-defined tasks. For instance, in [8], the performance of LLMs on understanding Turtle is evaluated across 6 different tasks (e.g., turtle syntax issue detection, SPARQL query generation, etc.) using the LLM-KG-benchmark³. Yet, the evaluation of some knowledge engineering tasks such as competency question generation or ontology evaluation present a significant research challenge due to factors such as the absence of standardized benchmarks or the inherent subjectivity in assessing correctness. For instance, in [1], the authors highlight that defining what constitutes a "good" competency question is a non-trivial task. Thus, they motivate the collection and comprehensive analysis of competency question requirements, to support the definition and creation of concrete fine-grained benchmark tasks and assessment dimensions.

The need to understand the strengths and limitations of different LLMs in domain-specific tasks has also become evident across disciplines. A *dimension-based evaluation strategy* for LLMs has already been adopted in various fields—examples of which we discuss next.

LLM Capability Assessment Methods Across Domains. To enable meaningful comparison between different LLMs and state-of-the-art approaches, collections of assessment tests have been developed. For instance, in the chemistry domain, the capabilities of different LLMs have been evaluated across a range of chemistry tasks [10]. The authors identify key capabilities critical to the domain, such as reasoning, understanding, and explaining. Subsequently, they assess each capability through a number of chemistry tasks. The comprehensive analysis enables researchers to select the best performing LLM (or alternative method) for a specific use case.

A similar approach is applied to systematically assess the reasoning skills of LLMs [33]. The authors utilize a collection of evaluation tests to examine deductive, inductive and abductive reasoning capabilities. Moreover, they define concrete assessment dimensions, relevant to the field, such such as answer correctness, explanation redundancy and completeness, etc. Such a fine-grained capability assessment allows the selection of the best-performing model for a concrete task while accounting for the desired output characteristics and model behavior.

²Wine Ontology - <https://www.w3.org/TR/owl-guide/wine.rdf>

³LLM-KG-Benchmark - <https://github.com/AKSW/LLM-KG-Bench>

In the field of psychology, researchers apply assessment tests, originally designed to evaluate non-cognitive human characteristics to understand traits exhibited by LLM [19]. The authors argue that uncovering the psychological profile, i.e., the beliefs, values and biases mimicked by a particular LLM can inform decisions about model selection and help mitigate potential negative real-world consequences.

Motivated by related work on LLM capability assessments in various fields, we previously proposed the development of a test benchmark for evaluating LLM skills in KE through multiple assessment dimensions [27]. In this paper, we extend this line of work, focusing on ontology evaluation and designing a series of test tasks that assess LLM capabilities in ontology issue detection, classification, explanation, and potential modeling correction. Moreover, to support the LLM capability assessment, we extend our previously presented student ontology corpus [11] by extracting and annotating a set of student-built ontology axioms to be used in experimental investigations of ontology evaluation implementations.

3. Ontology Evaluation Capability Assessment Tasks

The development of ontologies is a time-intensive process, and modeling errors are common—particularly when logical constraints are implemented and the work is carried out by novice knowledge engineers [22]. In addition, a recent survey highlights that ensuring ontology compliance with specified requirements is a key challenge, and that there is a significant lack of ontology testing tools [36]. Drawing on our experience in teaching students with varying technical backgrounds to create ontologies, we identify a set of core capabilities essential for supporting novice knowledge engineers in building and validating ontologies.

Concretely, we assess ontology modeling capabilities through four tasks, each designed to evaluate a specific competence. In Section 3.1, we define a task focused on evaluating the ability of LLMs to detect modeling issues. This is followed by a complementary task in Section 3.2, which assesses the classification of identified issues. Section 3.3 introduces a task targeting the ability to explain a modeling mistake and reason about its potential unintended consequences. Finally, Section 3.4 presents a constraint generation task, which evaluates the model’s ability to propose potential corrections to the identified modeling mistakes.

3.1. Task 1: Detecting an Ontology Modeling Issue

This task aims to assess the capability of LLMs to *detect a modeling mismatch* between a user’s intended constraint meaning I - expressed in natural language, and its formal representation modeled in an ontology axiom OA .

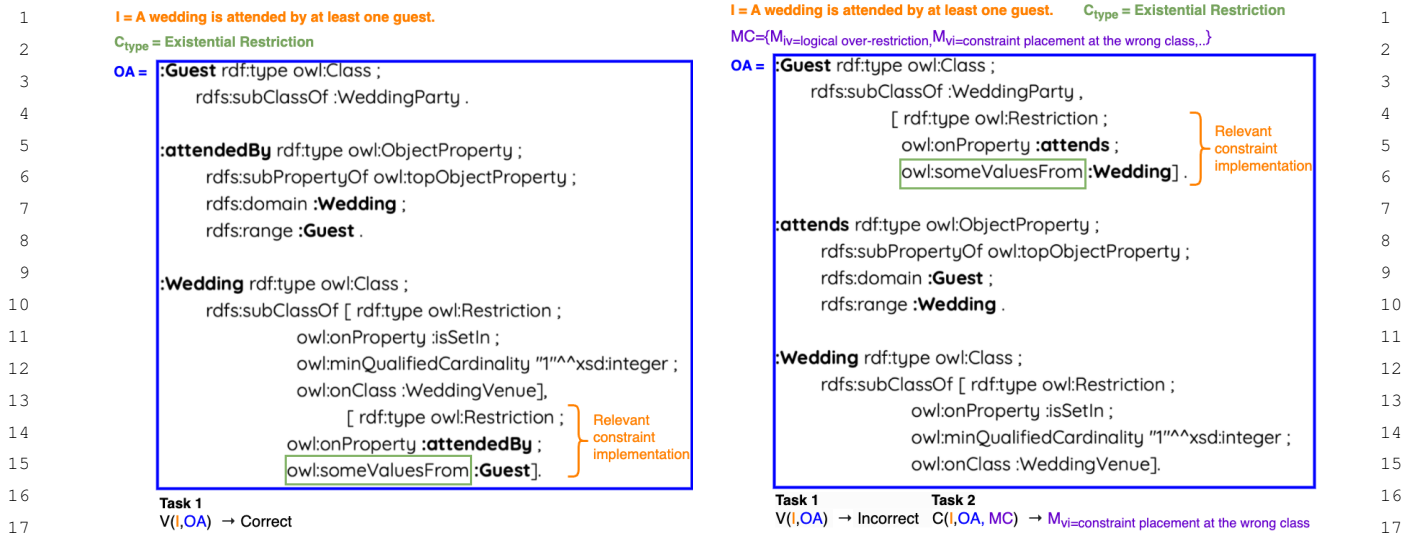
Inputs. Task inputs are the user-modeled ontology axiom OA represented in Turtle syntax and implementing a constraint type C_{type} , and the intention I , which constitutes the belief of what the modeling represents.

Goal. The goal of the task is to evaluate whether the modeled axiom OA correctly captures the expressed intention I . Note that OA may include several constraints of different types, each expressing a different restriction. Thus, the task also entails determining which of the modeled constraints is relevant to the intended meaning I .

Output. Expected output is the result of the binary validation $V(I, OA) \rightarrow \{Correct, Incorrect\}$, which returns *Correct* if the relevant C_{type} is correctly modeled within OA and OA semantically aligns with I and *Incorrect* otherwise.

Example. To exemplify the detection of modeling inconsistencies we consider the following case, displayed in Figure 1a: The intention I is described as "A wedding is attended by at least one guest.". The modeled OA includes two constraint. The relevant C_{type} to I is an *existential restriction (owl:someValuesFrom)*. In the given case, there is a semantic alignment between I and OA , thus the expected validation output for Task 1 is *Correct*.

Evaluation. The LLM capabilities on this task can be evaluated using standard classification metrics, including *accuracy, precision, recall, and F1-score* with respect to a manually evaluated gold standard.



(a) Example of a correctly modeled intention I within the ontology axiom OA containing a constraint type C_{type} , provided as input to Task 1.

(b) Example of an incorrectly modeled intention I within the ontology axiom OA containing a constraint type C_{type} , provided as input to Task 2.

Fig. 1. Visual representations of Task 1 (1a) and Task 2 (1b) exemplifying their corresponding inputs and outputs.

3.2. Task 2: Classifying an Ontology Modeling Issue

As a continuation from Task 1, this task's objective is to assess LLM capabilities in *identifying a modeling mistake type* M_{type} , given a user's intended constraint meaning I and an ontology axiom OA , containing at least one constraint C_{type} .

Inputs. Task inputs are an *incorrectly* modeled ontology axiom $OA_{incorrect}$ and the intention I , which constitutes the belief of what the modeling represents. Additionally a mistakes classification $MC = \{M_1, M_2, \dots, M_i\}$ is provided, where M_i is defined through a name $name(M_i)$ and a description $desc(M_i)$.

Goal. The goal of the task is to identify the mistake found in OA , according to the given mistake classification. It is important to note, that from the inputs, it is unclear where the mistake in $OA_{incorrect}$ is. Thus, the task entails determining which of the modeled constraints from OA is relevant to I and is incorrectly modeled. The task additionally tests the understanding and distinguishing between different ontology modeling mistake types.

Output. Expected output is the result of the classification $C(I, OA, MC) \rightarrow M_{type}^+$, $M_{type} \in MC$. The classification returns a non-empty set of one or more mistake types that best explain the mismatch between the intended meaning I and OA , considering a modeled C_{type} .

Example. To exemplify the detection of modeling inconsistencies we inject a modeling mistake in the ontology axiom from Figure 1a. We consider the following modified case, displayed in Figure 1b: the intention and relevant constraint type remain unchanged, i.e., $I = \text{"A wedding is attended by at least one guest."}$ and $C_{type} = \text{existential restriction}$. However, due to the change in the OA modeling, there is a semantic misalignment between I and its implementation. Since the constraint is added to the definition of the *Wedding* class, rather than the *Guest* class, the model is interpreted as "Every guest should attend at least one Wedding.", failing to capture the intended restriction from I . The expected output of Task 1 is *Incorrect*. As part of Task 2 the corresponding mistake is to be identified, given a set of known mistakes. The expected output in the described example is the classification of the mistake as $M_{vi} = \text{constraint placement at the wrong class}$, given the mistake types listed in Section 4.2.

Evaluation. The performance of LLMs on Task 2 can be assessed using standard multi-label classification metrics, including weighted *accuracy*, *precision*, *recall*, *F1-score* and *hamming loss*, i.e. the fraction of incorrectly predicted labels, with respect to a manually curated gold standard dataset.

3.3. Task 3: Explaining an Ontology Modeling Issue

While Task 1 and Task 2 test the capabilities of LLMs to identify and classify a mistake found in an ontology axiom, Task 3 assesses the LLM’s understanding of why a given axiom is incorrect— i.e., its capacity to explain the semantic mismatch between the intention I and the modeled axiom OA .

Inputs. Task inputs are a constraint intention I , an *incorrectly* modeled ontology axiom OA , as well as a mistake type M_i , found in OA , defined though a name $name(M_i)$ and a description $desc(M_i)$.

Goal. The objective of this task is to generate an explanation for why a mismatch exists between the intended meaning I and the modeled axiom OA . Specifically, the task involves understanding and explaining the semantic meaning of OA and identifying unintended inferences or outcomes that could arise from its misalignment with I .

Output. The expected task output is an explanation $E_{I,OA,M_i} = (I_{OA}, O_{unexp,I})$ of the occurrence of a mistake M_i in the context of an ontology axiom OA and an intention I , containing at a minimum I_{OA} , representing the intention modeled within OA and $O_{unexp,I}$, denoting possible unintended outcomes according to I .

Example. Provided the OA displayed in Figure 1b and the returned mistake type $M_{vi} = \text{constraint placement at the wrong class}$, an explanation should be generated. An example generated explanation is presented in Figure 2, where the parts corresponding I_{OA} and $O_{unexp,I}$ are annotated.

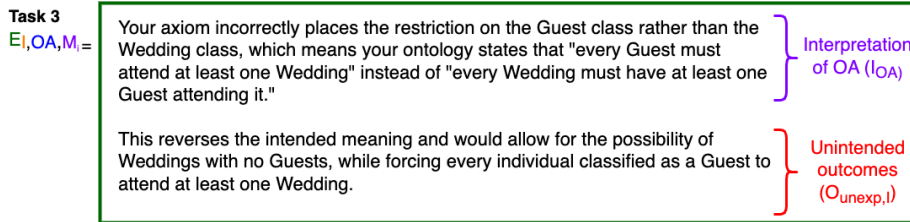


Fig. 2. Example explanation, outputted as result to Task 3, provided a constraint intention I , an ontology axiom OA , and a mistake type M_i .

Evaluation. The performance of LLMs on Task 3 can be assessed though an expert evaluation. We define an assessment schema to allow for a more objective dimension-based annotation. As fundamental quality aspects of the explanation to be assessed, we define *Modeling Comprehension*, *Inference Capability*, *Mistake Correction Competence* and *Vocabulary Usage*. For each dimension we assign a score s taking values between -1 (incorrect) and 1 (correct and complete) as detailed in Table 1.

This structured schema helps reduce subjectivity in expert judgments and ensures that different aspects of explanation quality are evaluated consistently. By capturing multiple dimensions of explanation quality, it also enabled the analysis of specific strengths and weaknesses of LLM-generated outputs.

3.4. Task 4: Generating a Correct(ed) Ontology Modeling

Task 4 extends the ontology modeling evaluation tasks from modeling issue detection, classification and explanation to the generation of an alternative modeling $OA_{alternative}$ as a potential correction of OA . As a preliminary, an axiom OA is to be stripped of the already included constraints to create the axiom base OA_{base} .

Inputs. The task takes as inputs a base ontology axiom OA_{base} , which is to be used as a starting point of the modeling; a constraint intention I which specified the intended meaning to be modeled in OA_{base} ; and C_{type} which determines the type of constraint to be included in the modeling.

Table 1

Expert-Assessment schema used to evaluate ontology axiom mistake explanations according to four key dimensions: *Modeling Comprehension*, *Inference Capability*, *Mistake Correction Competence* and *Appropriate Vocabulary Usage* and details on the assigned score s .

Quality Dimension	Score s
Modeling Comprehension	
The explanation contains a correct interpretation of the modeling of all relevant elements.	1
The explanation contains a correct interpretation of the modeling for a subset of the relevant elements.	0.5
The explanation does not contain an interpretation of the modeling.	0
The explanation contains an incorrect interpretation of the modeling.	-1
Inference Capability	
The explanation correctly and completely points out to (potential) unintended outcomes of the mistake.	1
The explanation correctly but partially points out to (potential) unintended outcomes of the mistake.	0.5
The explanation does not point out to (potential) unintended outcomes of the mistake.	0
The explanation incorrectly points out to (potential) unintended outcomes of the mistake.	-1
Mistake Correction Competence	
The explanation includes a correct modeling suggestion for fixing the mistake.	1
The explanation does not include a modeling suggestion for fixing the mistake.	0
The explanation includes an incorrect modeling suggestion for fixing the mistake	-1
Appropriate Vocabulary Usage	
The explanation demonstrates correct usage of domain-specific (ontology) terminology.	1
The explanation demonstrates incorrect usage of domain-specific (ontology) terminology.	-1

Goal. The task aims to assess the ability of LLMs to generate ontology modeling axioms with constraints, given a user intention I and a base ontology structure OA_{base} . This task not only evaluates the model's ability to generate a correct constraint but also tests its flexibility in modifying OA_{base} or C_{type} when the provided structure or constraint type are insufficient or inappropriate for modeling I correctly.

Output. The expected task output is the result of the generation $G(OA_{base}, I, C_{type}) \rightarrow OA_{alternative}$, where $OA_{alternative}$ should be represented in valid Turtle syntax.

Example. To exemplify the generation of an alternative constraint modeling, the OA from Figure 1b is stripped of all included constraints to generate OA_{base} , shown in Figure 3. Provided the given intention I and C_{type} to be used, the model is expected to generate an alternative modeling $OA_{alternative}$.

Evaluation. The evaluation of Task 4 relies on expert-evaluations to annotate the generated ontology axioms as correct or incorrect. Performance of LLMs can be assessed by calculating an *accuracy* score for each model with regards to the expert judgments.

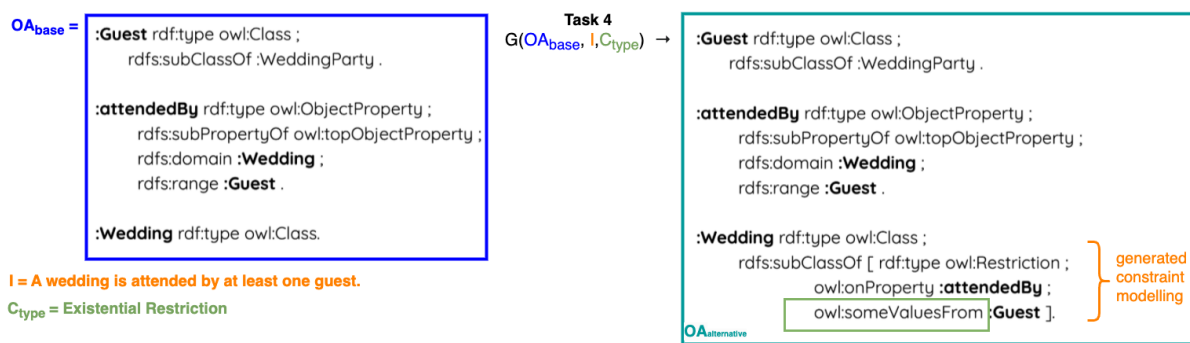


Fig. 3. Visual representations of Task 4 exemplifying the task inputs (OA_{base} , I , C_{type}) and output ($OA_{alternative}$).

4. Student Ontology Corpus: A Resource for Experimental Investigation in Knowledge Engineering

A large corpus of ontologies, along with their characteristics and documentation, is essential for developing and evaluating LLM-based solutions for KE tasks, such as those introduced in Section 3. To address this research need, we have compiled a collection of ontologies from student assignments submitted in knowledge engineering courses over the past decade. In the following sections, we provide an overview of this corpus (Section 4.1) and introduce a subset thereof used for the experimental investigations in this paper (Section 4.2).

4.1. Corpus Overview

In the following we describe the sources and extraction method used to create the corpus as well as the size and quality of the resulting collection of ontologies.

Sources. The corpus contains ontologies gathered from ontology development student projects from courses at two universities: Technische Universität Wien (TU Vienna) and the Vienna University of Economics and Business Administration (WU Vienna). The courses were held over a ten-year period from 2014 to 2024 and were offered to both undergraduates with little to no technical background (at WU Vienna) and graduate students specializing in Software Engineering, Data Science or related fields (at TU Vienna). The student assignments varied across the courses. In some instances, students were required to model a domain of their choice, while in others, they were provided with detailed ontology-based application descriptions and tasked with developing the necessary ontology accordingly. Additionally, students were required to formulate competency questions and document these and their ontology in a report, both of which were collected alongside each ontology in the corpus.

Extraction. To build the corpus, we employ a structured extraction workflow⁴, as described in detail in [11]. For each ontology, we analyze essential ontological properties such as the number of classes, properties, instances, and axioms. Furthermore, the analysis includes identifying error-prone structures using the method outlined in [21] and the Ontology Pitfall Scanner (OOPS!) [20].

Size. The corpus contains 585 ontologies from 19 different course instances. The ontologies vary in size, with beginner courses featuring smaller ontologies (averaging 26-27 classes and 250-300 axioms) and advanced courses leading to medium-sized ontologies (averaging 50 classes and 2400 axioms).

Quality. The corpus contains the original (ungraded) student submissions and the quality of the collected ontologies varies. Since the assignments were completed by both beginners in ontology modeling and students with a more technical background, some ontologies contain mistakes, while others are of very high quality. This variation makes the corpus a valuable resource for KE solutions, as it provides both positive and negative data examples, making it particularly useful for tasks such as ontology evaluation.

4.2. Expert-Annotated Subset Facilitating Ontology Evaluation Experiments

To support the experiments conducted in this paper, we selected a subset of ontologies from the corpus, extracted ontology axioms containing restrictions (universal, existential, and cardinality constraints), and performed an expert evaluation of their quality, as detailed in the following subsections.

4.2.1. Ontology Selection and Axiom Extraction

We selected ontologies built in the last year (2024) by undergraduate students with little to no technical background from the student ontology corpus. This subset was chosen for two reasons: 1) students with limited experience in ontology modeling are more likely to make mistakes, allowing for the natural collection of incorrect axioms without artificially introducing synthetic defects, and 2) the accompanying student reports provided detailed explanations of the intended meaning of each constraint (e.g., cardinality), enabling a more effective detection of modeling errors. The selection resulted in 31 ontologies covering topics from 16 different domains (e.g., Movies, Fashion, Law, etc.).

⁴The source code of the workflow is available at <https://github.com/wu-semsys/ontology-analysis> allowing others to utilize it for processing their own ontology collections

From the selected ontologies, we (manually) extracted an ontology axiom for each modeled and described existential (owl:someValuesFrom) or universal (owl:allValuesFrom) restriction, as well as for each (min/max/exact) cardinality constraint. An example axiom is shown in Figure 4. Each axiom is represented in Turtle⁵ and includes the definitions of the property, used for the restriction (*attendedBy* in Figure 4), the range of the restriction (*Guest*) as well as the restricted class (*Wedding*). In cases where multiple restrictions are applied to the domain or range class, all of them are extracted to ensure a complete representation of the resources. Similarly, any property characteristic (e.g., symmetry) assigned to the used property is included in the extracted axiom. For each axiom, we additionally extracted the constraint type and the student's modeling intention, as described in their report. For instance, the axiom from Figure 4 includes an existential restriction, described as: "Every wedding is attended by at least one guest." In total, we extracted 96 distinct ontology axioms and their associated modeling intentions (as text snippets).

4.2.2. Gold Standard for Mistake Detection and Classification.

To enable the creation of a gold standard, we conduct an expert-evaluation of the quality of each extracted ontology axiom. The assessments were completed by three researchers with experience in teaching KE courses to beginners. For each axiom, we assess 1) whether the student intention aligns with their modeled axiom; and 2) if a mismatch is detected, we identify the specific type of modeling mistake. Additionally, we collect further information on the clarity of the described student's intention and the meaningfulness of the constraint. This data will support follow-up studies on how these factors influence modeling quality.

Gold Standard Curation Methodology. To facilitate the piloting of the expert evaluation, 14 axioms (15%) were annotated by two experts each in an initial evaluation round. Following this, a discussion was held to clarify the axiom assessment criteria and establish a classification of mistake types. These 14 axioms were excluded from the gold standard and can be used to pilot experimental investigations.

For the remaining 82 axioms, we followed a three-step annotation process. First, each axiom was independently assessed by two experts. The inter-rater agreement between them was substantial, with a Fleiss' Kappa of 0.74 for the alignment between the student's intention and the modeled axiom, and 0.65 for the classification of mistake types. Second, axioms where the two annotators disagreed were reviewed by a third expert, and a majority vote was used to determine the final annotation. Third, to ensure the quality of the gold standard, cases where no majority could be reached regarding the modeling mistake, as well as selected axioms requiring further discussion, were reviewed in a meeting with all experts, and a final type agreement was then established for the gold standard value.

Gold Standard Overview and Contained Mistake Types. The final gold standard contains 82 ontology axioms, including 44 cardinality constraints, 25 existential restrictions, and 13 universal restrictions. 53 axioms were modeled correctly by the students, 3 axioms were correctly modeled under a concrete assumption, which was not clear from their intention, while the remaining 26 did not align with the student's intention and included at least one modeling mistake.

We identify six main mistake types present in the collected student ontologies which are related to the usage of cardinality, universal and existential restrictions:

- **Type I (6 axioms):** Mismatch between the domain and range of an object property and the constraint definition. Students often overlook the domain and range of object properties and thus define restrictions which do not comply with existing ontology definition.

```

:Guest rdf:type owl:Class ;
      rdfs:subClassOf :WeddingParty .

:attendedBy rdf:type owl:ObjectProperty ;
            rdfs:subPropertyOf owl:topObjectProperty ;
            rdfs:domain :Wedding ;
            rdfs:range :Guest .

:Wedding rdf:type owl:Class ;
         rdfs:subClassOf [ rdf:type owl:Restriction ;
                          owl:onProperty :attendedBy ;
                          owl:someValuesFrom :Guest ] .

```

Fig. 4. An ontology axiom, extracted from the definition of an existential constraint, restricting the class *Wedding*.

⁵Terse RDF Triple Language: <https://www.w3.org/TR/turtle/>

1 *Example:* Assuming the object property *takesPictureOf* is defined with a domain *Photographer* and range *Wedding*. To model the intention “A *Photographer* takes pictures of at least 10 *Guests*” one might add a constraint
2 “Photographer takesPictureOf min 10 Guest”. However, such a constraint could lead to an inconsistency or
3 wrong inferences such as *Guest* being a *sucClassOf Wedding*.
4

- 5 – **Type II (9 axioms):** Incorrect constraint placement of the constraint in the property range. Often novice knowl-
6 edge engineers define the constraint in the range of the object property rather than at the class-level, which
7 changes its logical interpretation and often leads to inconsistencies.

8 *Example:* To model the intention “A *Student* attends at least 3 *Courses*” one might add a constraint as the range
9 of the object property *attends* such as “attends min 3 *Course*”. However, this constraint placement would result
10 in the definition that the range of *attends* must be a *Thing* that attends 3 *Courses* rather than the intended
11 meaning.

- 12 – **Type III (4 axioms):** Logical over-restriction due to incorrect equivalence usage. A common student-made
13 mistake is the confusion between the usage of equivalence and *subClassOf* restrictions which potentially leads
14 to unintended over-restrictions.

15 *Example:* To model the intention “Every *FamousActor* plays a *MainCharacter*” one might define equivalence
16 between the class *FamousActor* and an *Actor*, who plays a *MainCharacter*. However, such a model additionally
17 introduces a restriction that every *Actor* who plays a *MainCharacter* is a *FamousActor*, which is typically not
18 intended.

- 19 – **Type IV (5 axioms):** Logical misunderstanding of the used constraint. Novice knowledge engineers might
20 choose a constraint type not fitted for modeling their intention. This often leads to a mismatch between their
21 ontology axiom and their intention. Examples of this mistake type are the incorrect usage of cardinality to
22 express value constraints and interpreting the universal or existential restrictions as exact cardinality of 1.

23 *Example:* To model the intention “A *Professor* may only work at one *University*” one might use a universal
24 restriction rather than an exact cardinality of 1. However, modeling the constraint using the universal restriction
25 would not restrict that a *Professor* only has one employment or that they must be employed at all.

- 26 – **Type V (4 axioms):** Logical under-restriction, caused by the trivial satisfaction of the universal restriction.
27 Novice knowledge engineers could incorrectly assume that the universal restriction implies the existential
28 constraint. This leads to valid instances with no property values at all when this was not intended.

29 *Example:* To model the intention “A *Professor* should work at a *University*” one might use a universal restric-
30 tion. However, the model would not restrict that a *Professor* must be employed at all. Combining the universal
31 restriction with an existential constraint or minimum cardinality prevents such cases.

- 32 – **Type VI (3 axioms):** Constraint placement at the wrong class. Sometimes novice knowledge engineers ignore
33 the direction of an object property and thus place the constraint at the wrong class, which changes its logical
34 interpretation and often leads to inconsistencies.

35 *Example:* To model the intention “At least 3 *Students* attend a *Course*” one might add a constraint to the class
36 *Student* such as “Student attends min 3 *Course*”. However, this modeling would correspond to the definition
37 that a *Student* must attend at least 3 *Courses* rather than the intended meaning.

- 38 – **Other (2 axioms)**

39 Although the dataset is imbalanced in terms of the number of axioms for each constraint type, or the examples of
40 each mistake type, our primary goal was to collect non-synthetic axiom examples where defects occurred naturally
41 during ontology development.
42

43 *Gold Standard Availability.* The created and annotated dataset is not published online to prevent potential data
44 leaks into the training data of large language models. However, interested researchers may request access to the
45 dataset.
46

47 5. Experimental Setup

48 With the conducted experimental investigations we aim to assess LLM capabilities in evaluating three specific
49 types of constraints: cardinality(min/max/exact), existential (owl:someValuesFrom), and universal (owl:allValuesFrom)
50
51

restrictions. To independently evaluate the capabilities of LLMs across the four tasks outlined in Section 3, we conduct four experiments utilizing the ontology corpus described in Section 4.2. The experiment design enables the comparison of both the overall performance of models and specific behaviors, enabling an informed LLM selection for future experimental investigations.

We begin by outlining the LLM selection and prompt design methodology in Section 5.1. We then proceed to detail each experiment individually in Section 5.2, addressing task-specific considerations such as piloting, concrete prompt implementations, and other relevant factors.

5.1. LLM Selection and Prompt Design Methodology.

For our experiments we utilize four LLMs from distinct model families: GPT 4o⁶ (gpt-4o-2024-08-06), Claude Sonnet 3.7⁷ (claude-3-7-sonnet-20250219), Llama 3.3⁸ (Llama-3.3-70B-Instruct-Turbo) and DeepSeek V3⁹. These LLMs represent a diverse sample of some of the most advanced options currently available, with a strong track record of high performance across diverse tasks in the KE field and across domains. For instance, the Turtle-understanding capabilities of various Claude and GPT version are demonstrated in [8]. Specifically GPT-4o and Claude Sonnet have qualified as experts in terms of their capabilities in ontology axiom validation surpassing smaller models [27] and previous work has highlighted the strengths of the GPT family in validating class membership relations [2]. We incorporate two open-source alternatives to support use cases requiring locally deployed solutions. Based on findings indicating that Llama 3 leads other open models in triple consistency validation [23], we selected its successor-Llama 3.3 for our investigations. Additionally, we include DeepSeek V3 in our selection, a model that has yet to be widely explored within the KE domain. Nonetheless, its predecessor—DeepSeek V2—has demonstrated intermediate skills in identifying ontology defects, matching the performance of Llama 3 [27].

Prompt Design and Strategy Selection. During the piloting stage of each experiment, our goal was to design prompts that produce high-quality outputs. We use the same 14 axioms that guided the creation of the gold standard annotations (see Section 4.2) to pilot our experiments, testing various prompting strategies and prompt variations.

We began with a simple prompt and, when necessary, refine it using specific prompting strategies identified in prior research as effective in improving accuracy and consistency. In particular, we explored the (RE2) technique introduced in [34], which encourages LLMs to re-read the original question before responding and has been shown to significantly improve reasoning capabilities by reducing errors caused by overlooked details and to enhance comprehension of complex tasks. Additionally, we consider the Plan-and-Solve (PS) [32] prompting technique which guides the model through systematic problem-solving phases, namely: understanding, planning, calculation, and conclusion.

5.2. Experiment Design

In the following, we describe each experiment, outlining its specific setup and evaluation approach. Additionally, in Table 2 we provide an overview of the carried out experiments summarizing main aspect such as dataset size and applied prompt techniques.

Experiment 1: Detecting an Ontology Modeling Issue. We experimented with various prompt phrasings during the pilot phase to identify the most effective approach for mistake detection. Based on these insights, we defined a prompt template—provided in Appendix A.1—which was used for the LLM-based evaluation of each axiom in the main experiment dataset. The prompt template follows a combination of the PS [32] and RE2 [34] prompting techniques and was completed by inserting the specific student intention along with their corresponding modeled axiom. From each LLM-generated output, a binary answer (*True/False*) was automatically extracted and compared to the gold standard annotation to assess model performance.

⁶GPT 4o - <https://openai.com/index/gpt-4o-system-card>

⁷Claude Sonnet - <https://www.anthropic.com/news/claude-3-7-sonnet>

⁸Llama 3.3 - https://www.llama.com/docs/model-cards-and-prompt-formats/llama3_3

⁹DeepSeek-V3 - <https://api-docs.deepseek.com/news/news1226>

Table 2

An overview of the experimental setup of the carried out experiments specifying for each experiment the addressed ontology evaluation task (according to Section 3), number of evaluated axioms ($N_{OntologyAxioms}$), applied prompting techniques, reference to the appendix section where the prompt template is provided and applied evaluation strategy.

Experiment ID	Task ID	$N_{OntologyAxioms}$	Prompting Techniques	Prompt Design Reference	Output Evaluation Strategy
Experiment 1	Task 1	82	PS, RE2	Appendix A.1	Gold Standard Comparison
Experiment 2	Task 2	29	PS, RE2	Appendix A.2	Gold Standard Comparison
Experiment 3	Task 3	29	-	Appendix A.3	Expert Assessment (custom)
Experiment 4	Task 4	82	RE2	Appendix A.4	Expert Assessment (binary)

Experiment 2: Classifying an Ontology Modeling Issue. Within this experiment, we aimed at classifying a modeling issue into a set of mistake types present in the experimental dataset. To determine the optimal approach for mistake type classification, we evaluated various prompting settings during the pilot phase: (1) Presenting each mistake type individually with its description in a separate independent prompt; (2) Presenting each mistake type individually with its description and a concrete example in a separate prompt; (3) Presenting all mistake types together in a single prompt with their corresponding descriptions; and (4) Presenting all mistake types together in a single prompt with their corresponding descriptions and concrete examples.

The piloting revealed that strategy (3), when combined with the PS [32] and RE2 [34] prompting techniques, resulted in the most accurate classifications while ensuring straightforward automated class extraction from the LLM outputs. This prompting approach provides the model with the complete set of mistake types simultaneously, enabling comparative analysis.

Based on these findings, we defined a prompt template for ontology axiom mistake classification (see Appendix A.2), which we applied to each incorrect axiom and axioms which hold only under a certain assumption not clearly specified in the student intention in the experiment dataset. From each LLM output, we extracted the identified mistake type and compared it against the expert-annotated gold standard to evaluate classification performance. In addition, we extracted the model-generated explanations to support further qualitative analysis in follow-up studies.

Experiment 3: Explaining an Ontology Modeling Issue. The experiment focused on designing an effective LLM-based implementation for explaining ontology modeling mistakes, emphasizing the interpretation of the modeled meaning and the identification of potential unintended outcomes. During the piloting stage, we explored different levels of explanation detail to ensure high quality of generated explanation and their feasible evaluation.

For the prompting setup, we investigated two approaches: (1) providing the model only with a description of the modeling mistake type, and (2) additionally including a concrete example of a mistake. While both methods yielded high-quality explanations, the second approach was ultimately chosen, as it produced more precise outputs that clearly conveyed the intended meaning and highlighted potential unintended consequences. Given the already high quality of the generated explanations, we did not incorporate the PS [32] or RE2 [34] techniques in this experiment. The final prompt template defined according to the findings from the piloting stage is available in Appendix A.3. Each request for a modeling mistake explanation, provides the model with a description of the present modeling issue and a concrete example, while constraining the explanation to 1-2 sentences for pedagogical effectiveness. Additionally, the model is asked to focus on possible unintended inferences, rather than suggesting mistake corrections.

Generated explanations were further processed to ensure they conform to the specified length constraint. Rather than resending prompts when responses exceeded the length constraint, our implementation automatically extracts only the first two sentences from any verbose response, ensuring conciseness without requiring additional API calls. To evaluate the results, each explanation was annotated through an expert-evaluation according to the assessment criteria in Table 1.

Experiment 4: Generating a Correct(ed) Ontology Modeling To enable the generation of alternate constraint modelings, especially in cases where a mistake was previously identified, we test LLMs' capabilities in generating correct axiom models given a specific constraint intention and a base ontology structure.

To facilitate Experiment 4, we begin by preparing the base axiom structures, removing any previously added constraints from the student axioms while preserving essential class and property definitions. For each prepared axiom, we construct a prompt contextualizing each case with the relevant domain, constraint type, and student intention available in the experiment dataset. During the piloting phase we observed that implementing the RE2 [34] technique improved generation quality as with the previous experiments. Thus, the final prompt template used integrates this strategy and is provided in Appendix A.4.

To ensure the syntactic validity of the generated constraint modelings, we implemented a validation pipeline using RDFLib¹⁰, which verifies the Turtle syntax of each axiom. When invalid outputs are detected, an iterative correction procedure is triggered, in which the model is provided with the exact error message returned by the validator and allowed up to three refinement attempts. This process enables us to distinguish between the models' initial generation capabilities and their capacity for self-correction when guided by explicit error feedback. Finally, all successfully generated axioms undergo expert review to evaluate their semantic accuracy and alignment with the intended constraints.

6. Experiment results

We evaluated the performance of four LLMs across a series of experiments targeting distinct ontology evaluation sub tasks. The results, presented in the following subsections, provide a comparative overview of model performance and offer insights into their respective strengths.

6.1. Experiment 1: Detecting an Ontology Modeling Issue

To assess the capabilities of LLMs in ontology evaluation—specifically in identifying mismatches between intended meaning and actual modeling—we compare the outputs from Task 1 against the ground truth annotations described in Section 4.2 (n=82). We report the accuracy, precision, recall, and F1 scores for each LLM in Table 3. Overall, DeepSeek V3 outperforms the other models, achieving 70.73% accuracy in ontology axiom evaluation and an F1 score of 65.71%, closely followed by Claude Sonnet. While GPT-4o matches DeepSeek V3 in accuracy and achieves the highest precision (55.56%) across all models, it records the lowest recall (38.46%). In contrast, both DeepSeek V3 and Claude Sonnet demonstrate significantly higher recall scores of 88.46%. Llama 3.3 demonstrates intermediate performance: although it records the lowest accuracy and precision scores among the evaluated models, it achieves a higher F1 score than GPT-4o.

While the LLMs demonstrated relatively low precision (ranging from 41.46% to 55.56%), those achieving high recall scores—such as DeepSeek V3 and Claude Sonnet—are well-suited for use in human-in-the-loop evaluation campaigns, where the goal is to identify potentially erroneous axioms for further assessment by human evaluators.

Table 3

LLM ontology evaluation capabilities in terms of identifying mismatches between intended meaning and actual modeling, assessed through accuracy, precision, recall and F1 scores.

Model	Accuracy	Precision	Recall	F1
GPT-4o	70.73%	55.56%	38.46%	45.4%
Claude Sonnet 3.7	69.51%	51.11%	88.46%	64.79%
DeepSeek V3	70.73%	52.27%	88.46%	65.71%
Llama 3.3 70B	59.76%	41.46%	65.38%	50.75%

¹⁰RDFLib- a Python package for RDF: <https://rdflib.readthedocs.io>

Performance on each constraint type. To gain a better understanding of each model’s performance on axioms involving different constraint types, we further analyze their results across each tested constraint (Figure 5). In terms of accuracy, we observe that some models are better suited for identifying mistakes in axioms with different constraint types. For instance, GPT-4o performs best in the evaluation of axioms containing existential and universal restrictions, achieving accuracy scores of 88% and 76.9%, respectively. In contrast, DeepSeek V3 demonstrates the highest accuracy in evaluating cardinality constraints, with a score of 70.5%. However, given the imbalanced nature of the ground truth data (with a disparity between correct and incorrect axiom examples), it is crucial to also consider the precision and recall scores of each model as well.

From the precision scores, we see that GPT-4o achieves 100% precision in evaluating axioms with universal constraints, outperforming other models, which show lower performance with scores below 50% in the same task. In contrast, the recall for evaluating universal constraints is highest (75%) when the DeepSeek V3 model is used. Similarly, Claude Sonnet, DeepSeek V3, and Llama 3.3 demonstrate perfect recall (100%) when evaluating existential restrictions, while GPT-4o showcases the best precision for this constraint type. Lastly, for cardinality constraints, Claude Sonnet achieves the highest recall (94.7%), while precision is highest (61.9%) for Llama 3.3. Looking at the F1 scores, DeepSeek V3 provides the best balance of recall and precision for both cardinality constraints (72.3%) and universal restrictions (50%), while GPT-4o performer best for existential restrictions (57.1%).

The differences among models and constraint types suggest that a hybrid workflow, leveraging each model’s strengths, could be highly effective. For instance, for identifying incorrectly used universal constraints, DeepSeek V3, which demonstrates high recall, can be employed to pre-select erroneous candidates for further assessment. These outputs can then be evaluated by GPT-4o, which shows perfect precision for this constraint type. The remaining axioms can be validated through a human-in-the-loop approach. While such a workflow would not fully automate the ontology evaluation process, it could potentially reduce manual effort while ensuring high evaluation quality.

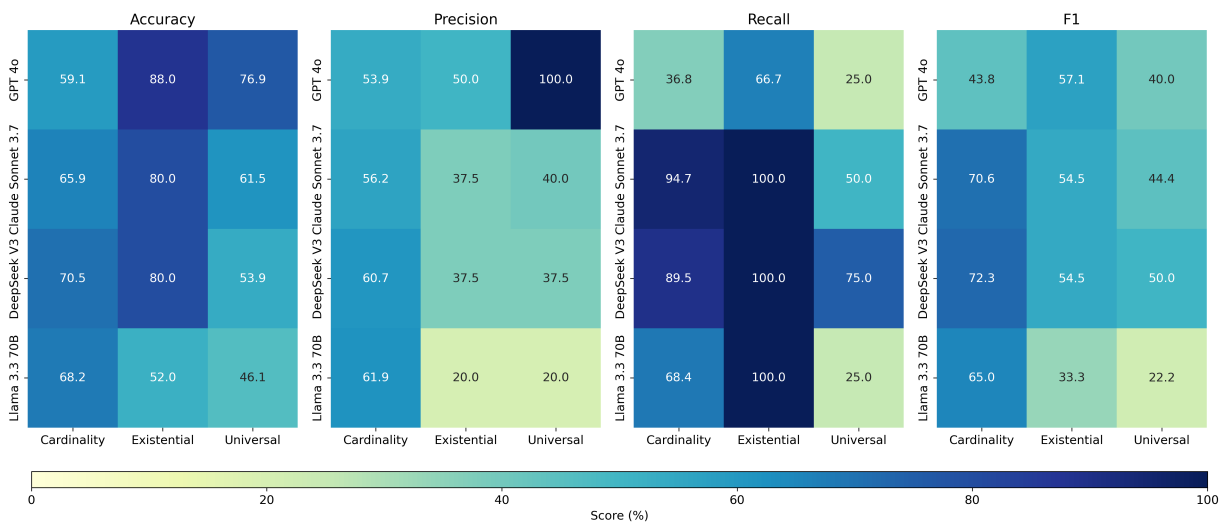


Fig. 5. Mistake identification capabilities in terms of accuracy, precision, recall and F1 scores for each LLM and constraint type.

6.2. Experiment 2: Classifying an Ontology Modeling Issue

Axioms annotated as incorrect in the ground truth or axioms, which include a mistake should a certain assumption not hold, ($n=29$) were used in an additional evaluation task, in which LLMs are tasked with classifying the specific mistake present in the constraint modeling. We compare the LLM mistake classification outputs against the ground truth mistake annotations by experts (see Section 4.2). For each model, we report the accuracy, weighted precision, recall and F1 scores as well as Hamming loss for the multi-label classification problem in Table 4. GPT-4o

outperforms the other models across all evaluated metrics, achieving 73.48% precision while minimizing misclassifications, with a Hamming loss of 0.09. Claude Sonnet and Llama 3.3 showcase similar performance, with precision scores of 56.57% and 51.61%, respectively, while DeepSeek V3 performs the worst, reaching only 39.8% precision.

Table 4

LLM ontology evaluation capabilities in terms of mistake type classification, assessed through accuracy, precision, recall, F1 and Hamming loss.

Model	Accuracy	Precision	Recall	F1	Hamming Loss
GPT-4o	65.52%	73.48%	66.67%	68.03%	0.09
Claude Sonnet 3.7	55.17%	56.57%	54.55%	54.87%	0.13
DeepSeek V3	48.28%	39.8%	45.45%	41.92%	0.16
Llama 3.3 70B	41.38%	51.61%	45.45%	43.97%	0.16

Performance on each mistake type. We further analyze the performance of each model in identifying specific modeling mistakes, particularly the mistake types described in Sect 4.2.2. Table 5 summarizes the results per model and mistake type providing insights into how well each LLM handles different modeling errors. Some mistake types, such as Type II—representing the incorrect placement of a constraint in the range of a property rather than as a class definition—are classified with high performance across all models, with F1 scores ranging from 88.9% by DeepSeek V3 to 100% by Claude Sonnet. However, the classification of other mistake types varies significantly between models. Type I mistakes—where the domain or range used for a constraint is inconsistent with the object property’s domain or range—are best handled by GPT-4o (F1 score: 80%), while other models perform substantially worse. Type III mistakes, which involve confusing equivalence with subclass constraint definitions, are best addressed by DeepSeek V3 (F1 score of 88.9%) while Llama 3.3 fails to classify this mistake type (F1 score of 0%). Similarly, for Type V, represented by and under-restriction, and type VI mistakes, where the restriction is applied to the wrong class, GPT-4o achieves the highest F1 score (74% and 80% respectively), while other models reach less than 60%. Finally, for mistake type IV (logical misunderstandings of the constraint) all models show overall lower performance. Llama 3.3 achieves the highest F1 score for this type reaching 42.9%. Additionally, two incorrectly modeled axioms from the ground truth that did not correspond to any predefined mistake category were nonetheless classified by all models into one of the existing categories.

Table 5

LLM mistake classification performance for each mistake type in terms of accuracy (Acc), precision (P), recall (R) and F1 scores. Additionally, for each mistake type the support (S) is provided.

Mistake		Performance Scores (%)															
Type	S	GPT 4o				Claude Sonnet 3.7				DeepSeek V3				Llama 3.3 70B			
		Acc	P	R	F1	Acc	P	R	F1	Acc	P	R	F1	Acc	P	R	F1
I	6	89.7	66.7	100	80	79.3	50	57.1	61.5	75.9	0	0	0	82.8	66.7	33.3	44.4
II	9	96.6	100	88.9	94.1	100	100	100	100	93.1	88.9	88.9	88.9	93.1	81.8	100	90
III	4	93.1	100	50	66.7	82.8	33.3	25	28.6	96.6	80	100	88.9	82.8	0	0	0
IV	5	75.9	25	20	22.2	79.3	33.3	20	25	82.8	0	0	0	72.4	33.3	60	42.9
V	4	93.1	75	75	75	89.7	66.7	50	57.1	82.8	40	50	44.4	89.7	100	25	40
VI	3	96.6	100	66.7	80	86.2	33.3	33.3	33.3	65.52	11.1	33.3	16.7	76.9	0	0	0
other	2	93.1	0	0	0	93.1	0	0	0	93.1	0	0	0	93.1	0	0	0

The variations in performance across models suggest that a combination of different LLMs could be leveraged to achieve improved classification results. For instance, an ensemble approach could be designed in which outputs from different models are weighted based on their demonstrated performance on specific mistake types. By assigning higher confidence to the predictions of models that have shown strength in identifying certain categories of modeling mistakes, the overall system could benefit from the complementary capabilities of each model, leading to more robust and accurate multi-label classification.

6.3. Experiment 3: Explaining an Ontology Modeling Issue

As a continuation of Tasks 1 and 2, the LLMs were further evaluated on their ability to explain modeling mistakes (n=29) by interpreting incorrect axioms and inferring potential unintended outcomes. To assess the LLM-generated explanations, we conducted an expert evaluation based on the criteria defined in Table 1. To ensure consensus on the assessment guidelines and address any open questions, 10% of the generated explanations were jointly evaluated by two experts during an in-person meeting. Subsequently, an additional 10% were assessed independently by both experts, yielding perfect inter-rater agreement (Fleiss' kappa = 1). To reduce manual effort, and given the high agreement achieved, the remaining explanations were evaluated by a single expert.

Explanations Quality Analysis. We analyze the model performance across four quality dimensions: *modeling comprehension*, *inference capability*, *mistake correction competence* and *vocabulary usage* and visualize the assessment results for each model in Figure 6. All models exhibit good-to-excellent *modeling comprehension* skills. Claude Sonnet demonstrates the best performance, providing perfect modeling interpretations for 28 axioms (97%). In one axiom containing two identified mistakes, however, Claude Sonnet only provided a relevant interpretation for one mistake, ignoring the second. Similarly, GPT-4o and DeepSeek V3 each offered a partial interpretation for one axiom. Overall, GPT-4o, DeepSeek V3 and Llama 3.3 generated correct axiom interpretations for 76-79% of the input axioms. GPT-4o failed to include a modeling explanation in 3 outputs (10%) and committed 3 mistakes in the remaining interpretations, while DeepSeek and Llama exhibited higher mistake rates, with errors in 5 (17%) and 7 (24%) cases, respectively.

In contrast, the models' *inference capabilities* are less reliable, demonstrating a higher number of incomplete outcome explanations and erroneous inferences. GPT-4o and Claude Sonnet produced incorrect inferences for the identified mistakes in the fewest cases (2 and 1, respectively). However, both had a high rate of missing inferences, failing to generate complete outcomes in 24% and 17% of cases, respectively. The weakest performance was observed in Llama 3.3, which generated complete inferences in only 11 cases (38%) while producing errors in 9 cases (31%). Overall, all models exhibited a tendency to omit important mistake outcomes that were highly relevant to the input intention, leading to correct but incomplete explanation generations.

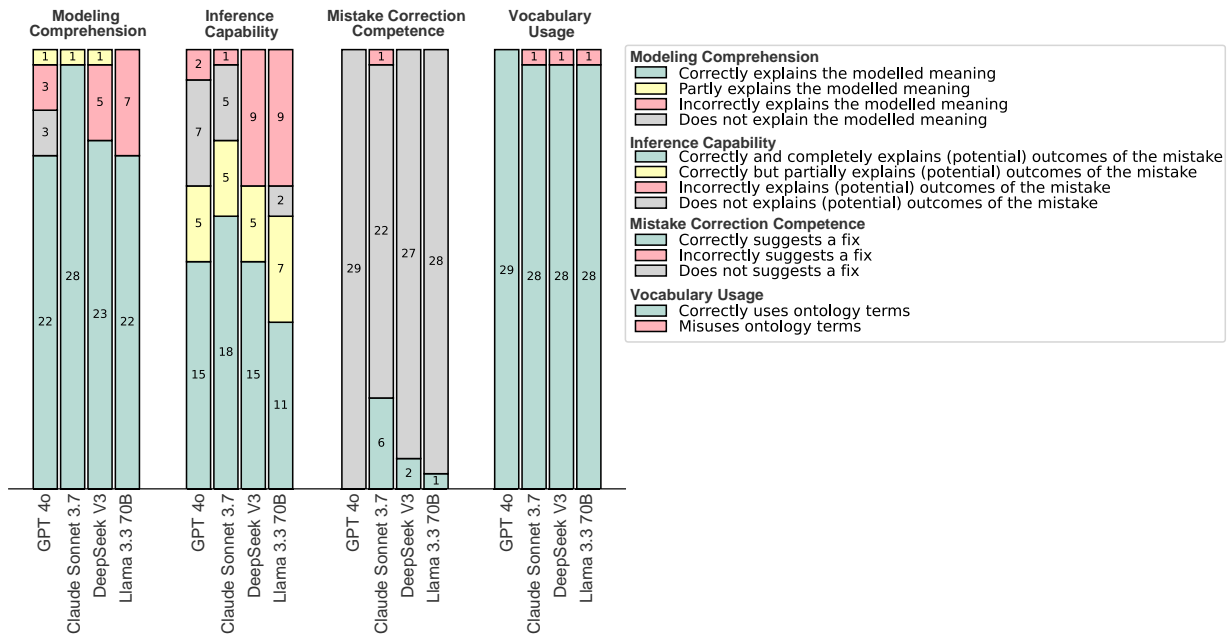


Fig. 6. Mistake explanation annotation frequencies for each model across 4 quality dimensions: modeling comprehension, inference capability, mistake correction competence and vocabulary usage.

Although the experimental setup and prompts were not explicitly designed to capture *mistake correction* suggestions in the generated explanations, we still annotated the corrections proposed by the models. As shown in Figure 6, most explanations did not include a mistake correction, as expected. Notably, GPT-4o strictly followed the prompt instructions and did not generate any unrequested corrections. In contrast, Claude Sonnet exhibited a tendency to propose modeling corrections, which were accurate for 6 axioms (21%) but incorrect in one instance. DeepSeek V3 and Llama 3.3 also occasionally included correct mistake corrections in their outputs.

Finally, we analyze the *usage of ontology-specific vocabulary* in the generated explanations. All models, particularly GPT-4o, demonstrate excellent vocabulary usage. Claude Sonnet, DeepSeek V3, and Llama 3.3 each make terminology-related mistakes in a single explanation.

Explanations Quality Scores. To quantitatively assess the models' mistake explanation capabilities and facilitate comparison, we compute a score for each model and quality dimension by summing the quality annotation scores according to Table 1. Additionally, we calculate a normalized overall score for mistake explanation performance, taking the sum of the modeling, inference, and vocabulary scores. We exclude the mistake correction dimension due to the specific experimental setup which did not ask for mistake correction suggestions.

Figure 7 visualizes a comparison of the models' performance. Across all models and dimensions, a consistent trend emerges: models that perform well in one dimension tend to maintain similar performance across the others. Claude Sonnet ranks highest among all models, followed by GPT-4o, DeepSeek V3, and Llama 3.3. An exception is the Vocabulary Usage dimension, where all models demonstrate similar capabilities, with GPT-4o achieving the highest score.

The experimental results suggest that while all tested LLMs demonstrate ontology-related skills in terms of accurate domain-specific vocabulary usage, they struggle with model interpretation and inference capabilities. This discrepancy is particularly concerning, as the correct use of terminology can create the illusion of expertise, potentially misleading users—especially less experienced ones, such as students—into trusting incorrect or flawed reasoning.

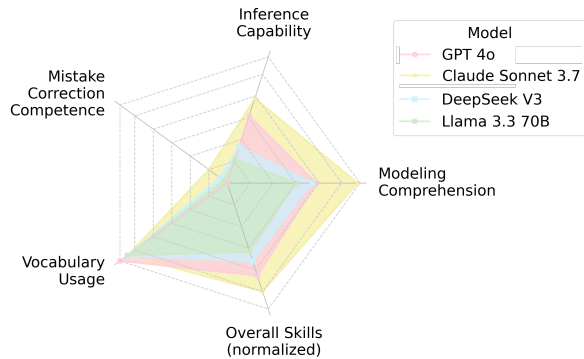


Fig. 7. Mistake explanation assessment scores of LLMs across distinct dimensions: modeling comprehension, inference capability, mistake correction competence, ontology-specific vocabulary usage, and a normalized value of the overall model skills.

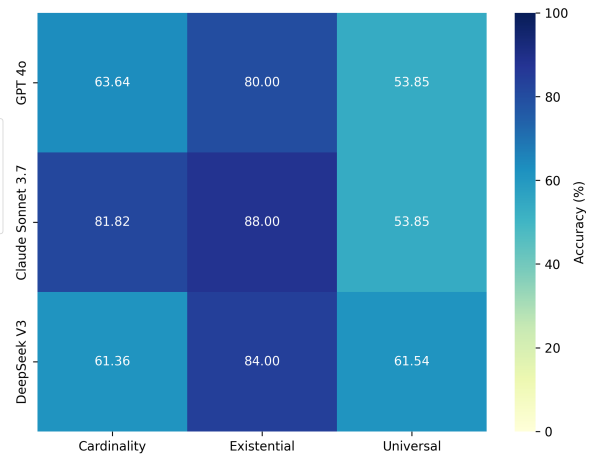


Fig. 8. Modeling accuracy across LLMs and constraint types.

6.4. Experiment 4 : Generating a Correct(ed) Ontology Modeling

We examine the capabilities of LLMs in generating ontology axioms containing universal, existential or cardinality constraints to determine whether they can be effectively integrated into an ontology generation workflow or an ontology evaluation pipeline. Of particular interest are the models' capabilities of generating alternative modeling

solutions to address erroneous modeling choices. We assess generated axioms for two aspects 1) syntactical validity of the produced (Turtle) axiom and 2) logical modeling correctness.

GPT-4o, Claude Sonnet and DeepSeek V3 successfully generated axioms in valid Turtle syntax. The models required at most one syntax refinement request (see Section 5) of the generated output for 27%, 15% and 39% of the axioms respectively. In contrast, Llama struggled significantly, failing to produce valid Turtle syntax for 85% of the axioms even after correction requests. Therefore, we exclude Llama-generated outputs from further analysis due to (1) the model's inability to produce results in the requested format and (2) the small sample size of syntactically valid axioms, which hinders meaningful analysis and comparison.

The axioms (n=82), generated by GPT-4o, Claude Sonnet and DeepSeek V3, were then evaluated for modeling correctness by two knowledge engineers. A randomly selected 10% subset of the LLM generated outputs was independently annotated by both experts, demonstrating substantial inter-rater agreement (Fleiss' kappa = 0.75). Any disagreements were resolved through discussion until a consensus was reached, and a classification framework for special cases was established. The remaining (90%) of the generated axioms were evaluated by a single expert to reduce manual annotation efforts.

We compute the accuracy of each model according to the expert evaluation. Claude Sonnet performs best across the tested models, achieving 79.27% accuracy of the generated axioms. GPT-4o and DeepSeek V3 performed significantly worse, achieving accuracy scores of 67.07% and 68.29%, respectively.

Accuracy based on Constraint Type. We investigate the capabilities of each model across different constraint types as visualized in Figure 8. It can be seen, that all assessed models produced high-quality modeling of existential constraints with modeling accuracy over 80%. Claude Sonnet outperforms other models in generating cardinality constraints, achieving an accuracy of 81.82%. In contrast, cardinality was modeled with significantly lower accuracy by GPT-4o and DeepSeek V3 with accuracy scores of 63.64% and 61.36% respectively. Universal restrictions proved to be the most challenging for the models to produce. GPT-4o and Claude Sonnet successfully implemented this constraint type in 53.85% of the cases, while DeepSeek V3 achieved the highest score with 61.54%.

Differences in LLM Behavior. To get further insights into the LLM capabilities, we perform an analysis of the behavior of each model and typical modeling mistakes they committed. Figure 9 provides an overview of the outcomes for each model and constraint type and we summaries the main behavior traits of the models below.

- **Introducing additional ontology elements.** GPT-4o adhered most closely to the input base axiom, ensuring that the provided classes and object properties were used within the generated constraint. In fact, the model introduced additional classes or properties only in approx 2.44% of the cases. Similarly, DeepSeek V3 added additional elements in 3.65% of the generated constraints. Claude Sonnet introduced new elements most frequently (for > 7% of the produced constraints). However, both DeepSeek V3 and Claude Sonnet defined new properties, which were not used within the constraints, in 2.44% of the generated axioms.
- **Generating incoherent and/or inconsistent constraints.** While GPT-4o and DeepSeek V3 used the provided input elements within the constraints, the models produced 8.54% and 7.32% incoherent and/or inconsistent constraints respectively. This occurred when the provided property was not defined in line with the student's intended constraint (e.g., property domain and range contradicted the intended meaning). In contrast, Claude Sonnet encountered such a mistake less frequently (in 2.44%, occurrence < 3 times).
- **Generating axioms with no added/enforced constraint.** A common mistake across models is the omission of a constraint in the generated axiom. GPT-4o exhibited this issue in 8.54% of the axioms, Claude Sonnet in 4.88% and DeepSeek V3 in 3.66%. In some cases a **cardinality constraint omission** can be explained by the fact that the ontology axiom already restricted the intention of the student. For example, if a property was defined as functional, ensuring a single value, and the student intended to add a maximum cardinality of 1, models may have intentionally omitted this addition. While such generated axioms align with the intention of the student, the requested constraint was not included, therefore such outputs were annotated as incorrect. However, for occurrences of **universal restriction omissions**, we could not identify a clear pattern to explain why the omission occurred. In 3.66% of cases, DeepSeek V3 would define the requested constraint, however, omit to link it to the class to be restricted, resulting in a **failure to enforce the constraint**.

- **Generating over-restrictive constraint axioms.** A recurring issue across models is the generation of over-restrictive cardinality constraints - in 8.54% of cases for GPT-4o, 3.66% for Claude Sonnet and 4.88% for DeepSeek V3. In some cases this was caused by the usage of equivalence rather than subClassOf restriction to model the constraint. For instance, when the student’s intention was "Card games must have at least 2 players", the constraint was modeled as "A card game is equivalent to a game with at least two players", unintentionally defining each game with at least two players as a card game. In several cases where the student’s intention included the keywords "exactly one", GPT-4o introduced both a universal constraint and an exact cardinality. However, this resulted in over-restrictive axioms such as "An event outfit should have exactly one pair of shoes and no other clothing item".
- **Modifying the requested constraint type.** In a single instance, Claude Sonnet changed the constraint type, which was requested in the prompt. This occurred because the student’s intention did not align with the constraint they intended to add. The generated axiom was annotated as correct since using the requested constraint type would have resulted in an incorrectly modeled intention.

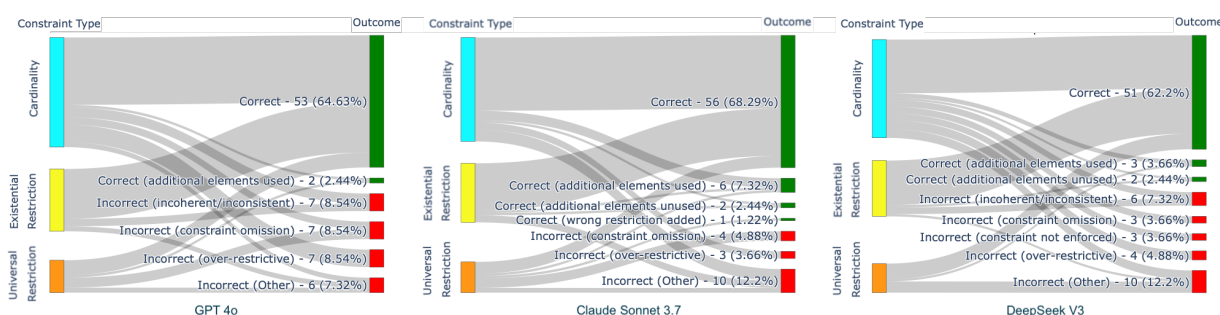


Fig. 9. A mapping between the generated constraint type (Cardinality, Existential, Universal) and typical outcomes produced by each model (GPT4o, Claude Sonnet and DeepSeek V3). Less frequent mistakes (occurrence < 3) are grouped into the category *Incorrect (Other)*.

To summarize, we observed varying behavioral features across the assessed models. The optimal model behavior depends on the task at hand. For constraint generation using a high-quality input ontology, a strict model might be the best choice to ensure alignment with the ontology definitions. However, when the input ontology is created by novice knowledge engineers and may contain issues, a model that can implement necessary improvements will help ensure the quality of the final constraints produced. We next look into each model’s performance based on the quality of the student implemented axiom.

Accuracy based on the quality of the student implemented axiom. It is essential to understand whether LLMs’ axiom generation capabilities vary depending on the task and conditions at hand. Therefore, we analyze the models’ performance by distinguishing between student axioms that were correctly implemented (before constraints were removed, see Section 5.2) and student axioms that were incorrectly modeled. The first case allows evaluating the models’ constraint generation capabilities when provided with high-quality input axioms. The second scenario enables assessing how well an LLM can support students in correcting their incorrect constraint implementations. Figure 10 visualizes the performance of each model on each constraint type for correct student axioms (left-hand side) and incorrectly modeled intentions (right-hand side).

We observe that the **inclusion of correctly modeled constraints into a high-quality base axiom** is a significantly simpler task, achieving accuracy between 66.67% and 96%. For cardinality constraints, Claude Sonnet performs best, achieving 96% accuracy, while existential restrictions are best modeled by DeepSeek V3, reaching 95.45% accuracy. Lastly, universal restrictions are handled well both by GPT-4o and DeepSeek V3, each achieving 77.78% accuracy. These findings suggest that a hybrid workflow could be designed, where different models are employed based on the specific constraint type to be added.

In the case of **adding constraints to previously incorrectly modeled student axioms**, the task becomes challenging, leading to lower accuracy (0-66.67%) across all models. Claude Sonnet outperforms other models for the

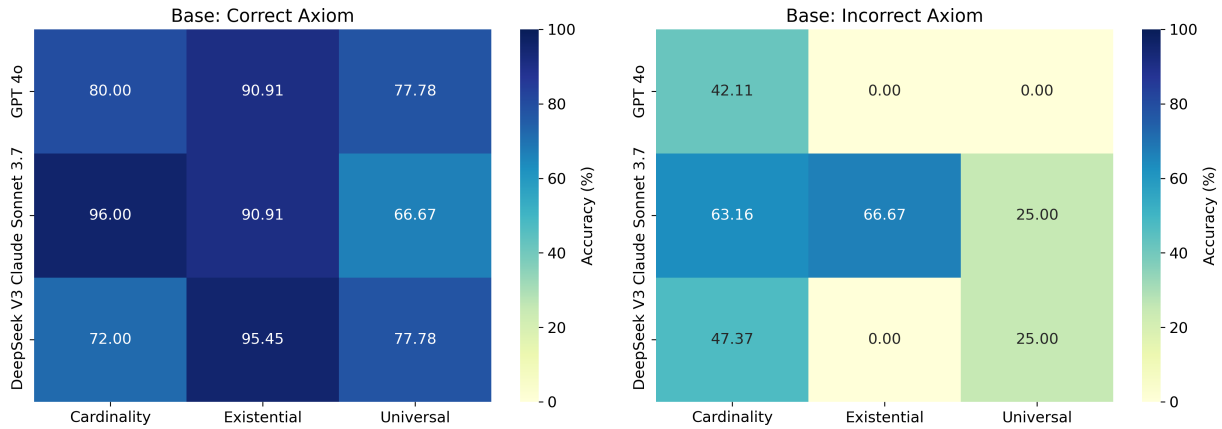


Fig. 10. Constraint modeling accuracy across LLM models and constraint types, distinguishing between student axiom quality.

inclusion of cardinality constraints and existential restrictions reaching accuracy of 63.16% and 66.67% respectively. However, universal restrictions are modeled with at most 25% accuracy when employing Claude Sonnet or DeepSeek V3 while GPT-4o completely fails this task (0% accuracy). These results indicate that LLMs are not yet capable at reliably generating alternative modeling as a correction of previous modeling mistakes, especially for the universal restriction.

7. Discussion

In this paper we conducted a series of experiments to evaluate the capabilities of large language models on a variety of ontology evaluation tasks.

Figure 11 provides a comparison of the models' performance on each evaluation task and highlights the strengths and weaknesses of each LLM, guiding future LLM selection for ontology-related experimental investigations. Below we summarize the main finding from the conducted experiments and offer recommendations for the usage of each assessed model.

Task 1: Detecting an Ontology Modeling Issue. Both DeepSeek V3 and Claude Sonnet outperform other models in terms of F1 scores on the task of detecting a modeling quality issue, given a constraint intention and its implementation in an ontology axiom. Nevertheless as reported in Section 6.1, the error detection precision of the models is significantly lower than their recall score. This indicates that these models are best suited at identifying potential defect candidates and can be included in initial stages of multi-phase ontology evaluation workflows which combine diverse validation techniques and actors. For instance, the models can be applied in an initial stage of human-in-the-loop evaluation campaigns in order to narrow down parts of the ontology where human review is required.

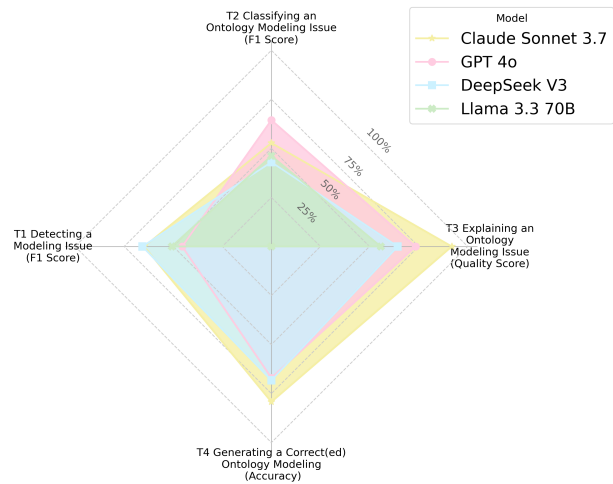


Fig. 11. LLM ontology evaluation capabilities across evaluation sub-tasks: detecting and classifying a modeling issue (T1, T2), evaluated using F1 scores; explaining a modeling issue (T3), assessed via a normalized quality score as defined in Section 3.3 and Section 6.3; and constraint modeling generation (T4) measured using accuracy.

Task 2: Classifying an Ontology Modeling Issue. GPT-4o demonstrates superior performance in classifying ontology modeling issues when provided with an incorrect ontology axiom. This makes GPT-4o particularly well-suited for workflows where a mistake candidate is previously identified through other techniques. In addition, the analysis in Section 6.2 revealed that although GPT-4o achieves the best overall F1 score, each LLM demonstrates unique strengths in identifying specific types of mistakes. As a result, constructing hybrid workflows that incorporate multiple models, each selected for its particular strengths, holds great potential for improving overall results.

Task 3: Explaining an Ontology Modeling Issue. Claude Sonnet demonstrates the best performance in generating high-quality explanations of modeling mistakes, correctly interpreting the meaning of provided erroneous axioms and inferring unintended outcomes caused by a modeling issue. Given its performance, Claude Sonnet is highly recommended for scenarios that prioritize error explanation, especially in the context of supporting novice knowledge engineers in understanding ontology constraints and common errors.

Task 4 : Generating a Correct(ed) Ontology Modeling. Claude Sonnet demonstrates the best accuracy on the task of generating a constraint modeling according to a provided intention and a base ontology axiom. The analysis in Section 6.4 further revealed distinct model behaviors, particularly in how each model handles the task. While GPT-4o tends to follow provided instructions strictly, Claude Sonnet showcased a remarkable flexibility. Claude’s ability to adapt to the problem at hand allowed it to successfully generate accurate constraint models, even in cases where the provided axiom was incomplete or needed to be adjusted for alignment with the specified constraint. Therefore, while ontology generation workflows might benefit from the strengths of each of the three models—DeepSeek V3, GPT-4o, and Claude Sonnet, Claude Sonnet is the clear choice for ontology evaluation workflows where a constraint generation is, for instance, suggested as a mistake correction.

Overall, for future ontology evaluation campaigns relying on LLMs, if one LLM is to be selected, Claude Sonnet is the clear choice, as it performs best across various tasks in ontology evaluation. However, employing more cost-efficient models whenever possible in this workflow can help balance performance and resource usage. Moreover, a hybrid workflow that combines multiple models—such as DeepSeek V3 for mistake candidate detection, GPT-4o for mistake classification, and Claude Sonnet for explanation and correct(ed) constraint generation—would provide an optimal approach.

8. Conclusion and Outlook

Despite the growing interest in leveraging LLMs for Knowledge Engineering tasks, significant challenges persist. These include the lack of systematic evaluation across diverse domains—particularly in the context of ontology evaluation (P1), limited comparative analysis of different LLMs (P2), the absence of tailored benchmarks (P3), and difficulties in objectively assessing LLM outputs (P4).

Contributions and Main Findings. To address these gaps we investigate how LLMs can support ontology evaluation. We investigate LLM capabilities in four sub-tasks tasks: *detecting*, *classifying*, *explaining* a modeling issue and *correcting* it by generating an alternative modeling of an incorrect implementation (RQ1). To this end, we formally define these four ontology evaluation sub-tasks and perform four experiment assessing the capabilities of various LLMs (GPT-4o, Claude Sonnet, DeepSeek V3, and Llama 3.3) on them. Within this capability assessment we explore strengths and limitations of each LLM in different ontology evaluation aspects (RQ2) and investigate persistent model traits in how they approach each task (RQ3). Our findings reveal model-specific strengths, behaviors, and the suitability of each model for different ontology evaluation tasks. DeepSeek V3 outperforms other models in defect detection, achieving a 65.71% F1 score, while GPT-4o excels in defect classification with a 68.03% F1 score. Claude Sonnet generates the best explanations, reaching 91.95% of the total points according to our custom criteria, and also produces the most accurate alternative constraint modelings with 79.27% accuracy. Additionally, we observe distinct model behaviors in terms of following instructions: GPT-4o strictly preforms the task as instructed, while Claude Sonnet demonstrates a higher flexibility in its task interpretation. These findings highlight the importance of selecting the most suitable LLM based on the specific task requirements.

To support the experimental investigations, we additionally construct and annotate a benchmark dataset containing student-built ontologies, which can be utilized for follow-up experimental investigation of various KE tasks.

Moreover, we explore how an objective LLM output assessment can be carried out to guide expert evaluations whenever a gold standard comparison is not feasible (RQ4). We introduce an annotation schema for the qualitative assessment of ontology mistake explanation outputs and show its effectiveness in generating reliable requirement-based expert judgments. We believe this approach can be adopted to create an assessment schema for annotating LLM-generated outputs of various tasks both in the KE domain and other areas.

Limitations and Future Work. While our study provides insights into the capabilities of LLMs for ontology evaluation, it is limited to a specific setting: the detection, explanation, and correction of ontology modeling issues related to existential, universal, and cardinality constraints. As a result, the findings may not fully generalize to other modeling aspects. Future work will extend the evaluation to include property characteristics such as symmetry and transitivity, enabling a broader assessment of LLM capabilities in ontology evaluation. Additionally, we aim to broaden the range of tested models by including a larger and more diverse set of LLMs, particularly those that are less commonly explored in current research. Additionally, the conducted experiments produced supplementary LLM outputs, such as the reasoning provided by models when detecting defects in ontology axioms. Future work will analyze these unguided explanations and compare them with the guided explanations from the dedicated explanation task, with the goal of gaining deeper insights into the reasoning strategies, consistency, and explanatory capabilities of different LLMs.

Outlook and Application. Looking ahead, the findings from this study can inform the development of intelligent applications that support students or junior knowledge engineers in building and validating ontologies. By leveraging the strengths of different LLMs across specific ontology evaluation tasks, we envision the creation of interactive educational tools that assist learners in identifying, understanding, and correcting modeling issues. These tools could integrate multi-step workflows that combine several models—each optimized for a particular sub-task such as defect detection, explanation, or correction—thereby offering targeted feedback and guidance throughout the ontology development process. Such systems hold promise for enhancing ontology education and making knowledge engineering more accessible and effective.

CRediT authorship contribution statement

Stefani Tsaneva: Conceptualization, Methodology, Validation, Formal analysis, Investigation, Resources, Data Curation, Writing - Original Draft, Writing - Review & Editing, Visualization, Project administration. **Guntur Budi Herwanto:** Conceptualization, Methodology, Software, Validation, Formal analysis, Investigation, Writing - Original Draft, Writing - Review & Editing. **Majlinda Llugiqi:** Conceptualization, Resources, Data Curation. **Marta Sabou:** Conceptualization, Writing - Original Draft, Writing - Review & Editing, Supervision, Funding acquisition, Project administration.

Acknowledgments

This research was funded in whole or in part by the Austrian Science Fund (FWF) BILAI (10.55776/COE12) and HOnEst (V 745) projects. For open access purposes, the author has applied a CC BY public copyright license to any author accepted manuscript version arising from this submission. Additionally, the work was supported by the PERKS (101120323) project, co-funded by the European Union. Views and opinions expressed are, however, those of the authors only and do not necessarily reflect those of the European Union. Neither the European Union nor the granting authority can be held responsible for them.

Appendix A. LLM Prompts used in the carried out experiments

In this section we provide concrete implementations of the four ontology evaluation tasks defined in Section 3. Concretely, we provide the constructed LLM prompts used in the experiments described in this paper (Section 5).

A.1. LLM Prompt for Experiment 1 (Detecting an Ontology Modeling Issue)

The prompt template for detecting a modeling issue is defined below and is to be completed by filling in a concrete constraint intention I and an ontology axiom OA , where this intention is modeled.

Is the intention correctly modelled in the ontology axiom? Answer Yes/No and explain your choice concisely.

Intention: { I }
Axiom: { OA }

Read the question again

Is the intention correctly modelled in the ontology axiom?

Intention: { I }
Axiom: { OA }

Let's first understand the problem and devise a plan to solve it.
Then, let's execute the plan step by step and provide the final answer.

[Problem Understanding]:

- Understand what the intention is trying to model.
- Analyze how the axiom attempts to represent it.

[Plan]:

- Determine the correctness of the axiom by evaluating constraints and placements.

[Solving/Calculations]:

- Analyze in detail if the axiom truly captures the intended meaning.
- Consider common mistakes like trivial satisfaction, open-world assumptions, or logical misunderstandings.

[Answer]:

- Provide a final Yes/No answer with a concise explanation.

Answer: [Yes/No]

Explanation: [Your explanation]

A.2. LLM Prompt for Experiment 2 (Classifying an Ontology Modeling Issue)

The mistake classification prompt template used for the experimental investigation of mistake type classification is shown below. Similarly to the template for Task 1 (mistake detection), the prompt is to be completed with a concrete constraint intention I and a corresponding ontology axiom OA .

Consider the following types of modeling mistakes:

1. Logical under-restriction, caused by the trivial satisfaction of the universal restriction: Often novice ontology engineers incorrectly assume that the universal restriction implies the existential constraint. This leads to instances with no property values at all to be valid when this was not intended.
2. Logical under-restriction, caused by the Open World Assumption: Often novice ontology engineers incorrectly assume that unmodelled relations would be invalid. This leads to instances with property values that were not intended.

3. Logical misunderstanding of the used constraint: Often novice ontology engineers choose the wrong constraint type to model their intention which leads to mismatch between their ontology axiom and their intention.
4. Logical over-restriction due to incorrect equivalence usage: Often novice ontology engineers confuse the usage of equivalence and subclassOf restrictions which leads to unintended over-restrictions.
5. Incorrect constraint placement of the constraint in the property range: Often novice ontology engineers define the constraint in the range of the object property rather than at the class-level, which changes its logical interpretation and often leads to inconsistencies.
6. Constraint placement at the wrong class: Often novice ontology engineers ignore the direction of the object property and thus place the constraint at the wrong class, which changes its logical interpretation and often leads to inconsistencies.
7. Mismatch of the domain and range of the property and the constraint usage: Often novice ontology engineers overlook the domain and range of object properties and thus define restrictions which do not comply with the ontology.

Given this case:

Intention: {I}

Axiom: {OA}

Assuming there is a modeling error, which of the mistake types best describes it?

Read the question again:

Which of these mistake types best describes the modeling error?

Intention: {I}

Axiom: {OA}

Let's first understand the problem and devise a plan to solve the problem. Then, let's carry out the plan, solve the problem step by step, and give the ultimate answer. Please explicitly generate the mentioned process:

[Problem Understanding]: Understand the intention, the axiom, and the potential modeling mistakes that could occur.

[Plan]: Systematically evaluate each mistake type to determine which one best matches the error in the axiom.

[Solving/Calculations]: Analyze the axiom in detail, comparing it against each mistake type to find the best match.

[Answer]: Identify which mistake type best describes the error and explain why.

Format your response as:

Mistake Type: [Type name]

Explanation : [Your explanation]

A.3. LLM Prompt for Experiment 3 (Explaining an Ontology Modeling Issue)

The prompt template used for the experimental investigation of ontology mistake explanations is shown below. The prompt is to be completed with several elements: a constraint intention I , its corresponding ontology axiom OA , a concrete mistake type M_i detected in the axiom represented through a name $name(M_i)$, a description $desc(M_i)$ of the mistake and a concrete example of its occurrence $example(M_i)$.

You are an expert in ontology engineering and semantic web technologies. Given the following:

1. A student's explanation of what they intended to model
2. The actual ontology axiom they created
3. The type of error identified in their modeling
4. A generic explanation of that error type: {name(M_i)}: {desc(M_i)}
5. An example of this type of error: {example(M_i)}

Please provide a concise explanation in EXACTLY 1-2 sentences to the student of why their axiom is modelled incorrectly (e.g., what unintended inferences could be made). Do not suggest corrections - focus only on explaining the error.

Student's intended meaning: {I}

Student's axiom: {OA}

Error type: {M_i}

A.4. LLM Prompt for Experiment 4 (Generating a Correct(ed) Ontology Modeling)

The prompt template used in the experimental assessment of LLM capabilities in generating correct ontology axioms including constraints is provided below. The prompt is to be completed with a specific intention I of what should be constrained in the context of a domain $domain(I)$, a constraint type C_{type} to be used and a base axiom OA_{base} to which the constraint should be added.

Task: Generate a complementary Turtle syntax based on this context:

Base axiom: {OA_base}

Domain: {domain(I)}

Constraint Type: {C_type}

Intention: {I}

Requirements:

1. Align with the given context
2. Be syntactically valid
3. Use minimal statements
4. Combine with base axiom

Read the task again

Task: Generate a complementary Turtle syntax based on this context:

Base axiom: {OA_base}

Domain: {domain(I)}

Constraint Type: {C_type}

Intention: {I}

Requirements:

1. Align with the given context
2. Be syntactically valid
3. Use minimal statements
4. Combine with base axiom

Now that you've read it twice, please:

```

1 1. Generate valid Turtle syntax that complements the base axiom
2 2. Ensure your output maintains semantic consistency with the original intent
3 3. Verify that the combined result (base + new) forms a coherent ontological
4   constraint
5
6 Your output should combine the base axiom with your additions in a complete, valid
7   Turtle document.
8
9

```

References

- [1] R. Alharbi, J. de Berardinis, F. Grasso, T.R. Payne and V. Tamma, Characteristics and Desiderata for Competency Question Benchmarks, in: *Proceedings of the Special Session on Harmonising Generative AI and Semantic Web Technologies (HGAIS 2024) co-located with the 23rd International Semantic Web Conference (ISWC 2024), November 13, 2024, Baltimore, Maryland*, CEUR-WS, 2024.
- [2] B.P. Allen and P.T. Groth, Evaluating Class Membership Relations in Knowledge Graphs using Large Language Models, in: *The Semantic Web: ESWC 2024 Satellite Events*, Springer Nature Switzerland, Cham, 2025, pp. 14–24. doi:10.1007/978-3-031-78952-6_2.
- [3] B.P. Allen, L. Stork and P. Groth, Knowledge Engineering using Large Language Models, *arXiv preprint arXiv:2310.00637* (2023).
- [4] S. Carta, A. Giuliani, L. Piano, A.S. Podda, L. Pompianu and S.G. Tiddia, Iterative zero-shot LLM prompting for knowledge graph construction, *arXiv preprint arXiv:2307.01128* (2023).
- [5] C.-H. Chiang and H.-y. Lee, Can Large Language Models Be an Alternative to Human Evaluations?, in: *Proceedings of the 61st Annual Meeting of the Association for Computational Linguistics*, 2023. doi:10.18653/v1/2023.acl-long.870.
- [6] N. Fathallah, S. Staab and A. Algergawy, LLMs4Life: Large Language Models for Ontology Learning in Life Sciences, *arXiv preprint arXiv:2412.02035* (2024).
- [7] N. Fathallah, A. Das, S. De Giorgis, A. Poltronieri, P. Haase and L. Kovriguina, Neon-GPT: a large language model-powered pipeline for ontology learning, in: *The Semantic Web: ESWC 2024 Satellite Events*, Springer Nature Switzerland, Cham, 2025, pp. 36–50. doi:10.1007/978-3-031-78952-6_4.
- [8] J. Frey, L.-P. Meyer, F. Brei, S. Gründer-Fahrer and M. Martin, Assessing the evolution of llm capabilities for knowledge graph engineering in 2023, in: *The Semantic Web: ESWC 2024 Satellite Events*, Springer Nature Switzerland, Cham, 2025, pp. 51–60. doi:10.1007/978-3-031-78952-6_5.
- [9] D. Garijo, M. Poveda-Villalón, E. Amador-Domínguez, Z. Wang, R. García-Castro and O. Corcho, LLMs for Ontology Engineering: A landscape of Tasks and Benchmarking challenges, in: *Proceedings of the Special Session on Harmonising Generative AI and Semantic Web Technologies (HGAIS 2024) co-located with the 23rd International Semantic Web Conference (ISWC 2024), November 13, 2024, Baltimore, Maryland*, CEUR-WS, 2024.
- [10] T. Guo, B. Nan, Z. Liang, Z. Guo, N. Chawla, O. Wiest, X. Zhang et al., What can large language models do in chemistry? a comprehensive benchmark on eight tasks, *Advances in Neural Information Processing Systems* **36** (2023), 59662–59688.
- [11] G. Herwanto, S. Tsaneva and M. Sabou, Ontology Corpora for LLM-based Knowledge Engineering Research, in: *Proceedings of the Special Session on Harmonising Generative AI and Semantic Web Technologies (HGAIS 2024) co-located with the 23rd International Semantic Web Conference (ISWC 2024), November 13, 2024, Baltimore, Maryland*, CEUR-WS, 2024.
- [12] H. Khorashadizadeh, N. Mihindukulasooriya, S. Tiwari, J. Groppe and S. Groppe, Exploring In-Context Learning Capabilities of Foundation Models for Generating Knowledge Graphs from Text, in: *TEXT2KG 2023: Second Int. Workshop on Knowledge Graph Generation from Text, May 28 - Jun 1, 2023, co-located with Extended Semantic Web Conference (ESWC), Hersonissos, Greece*, CEUR-WS, 2023.
- [13] M. Llugiqi, F.J. Ekaputra and M. Sabou, From Experts to LLMs: Evaluating the Quality of Automatically Generated Ontologies, in: *2nd Workshop on Evaluation of Language Models in Knowledge Engineering (ELMKE), co-located with ESWC-25*, to appear.
- [14] P. Mateiu and A. Groza, Ontology engineering with large language models, in: *2023 25th International Symposium on Symbolic and Numeric Algorithms for Scientific Computing (SYNASC)*, IEEE, 2023, pp. 226–229.
- [15] F. Neuhaus, Ontologies in the Era of Large Language Models? a Perspective, *Applied ontology* **18**(4) (2023), 399–407. doi:10.3233/ao-230072.
- [16] N.F. Noy, D.L. McGuinness et al., Ontology development 101: A guide to creating your first ontology, Stanford knowledge systems laboratory technical report KSL-01-05 and Stanford Medical Informatics Technical Report SMI-2001-0880, 2001.
- [17] N. Noy, Y. Gao, A. Jain, A. Narayanan, A. Patterson and J. Taylor, Industry-scale knowledge graphs: lessons and challenges, *Commun. ACM* **62**(8) (2019), 36–43. doi:10.1145/3331166.
- [18] S. Pan, L. Luo, Y. Wang, C. Chen, J. Wang and X. Wu, Unifying large language models and knowledge graphs: A roadmap, *IEEE Transactions on Knowledge and Data Engineering* (2024).
- [19] M. Pellert, C.M. Lechner, C. Wagner, B. Rammstedt and M. Strohmaier, AI Psychometrics: Assessing the Psychological Profiles of Large Language Models Through Psychometric Inventories, *Perspectives on Psychological Science* **19**(5) (2024), 808–826. doi:10.1177/17456916231214460.
- [20] M. Poveda-Villalón, A. Gómez-Pérez and 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**(2) (2014), 7–34. doi:10.4018/ijswis.2014040102.

- [21] A. Prock, Hybrid Human-Machine ontology verification: Identifying common errors in ontologies by integrating human computation with ontology reasoners, Master's thesis, TU Vienna, 2021.
- [22] A. Rector, N. Drummond, M. Horridge, J. Rogers, H. Knublauch, R. Stevens, H. Wang and C. Wroe, OWL pizzas: Practical experience of teaching OWL-DL: Common errors & common patterns, *Lecture Notes in Artificial Intelligence (Subseries of Lecture Notes in Computer Science)* **3257** (2004), 63–81. ISBN 978-3-540-23340-4. doi:10.1007/978-3-540-30202-5_5.
- [23] A.G. Regino and J.C. Dos Reis, Can LLMs be Knowledge Graph Curators for Validating Triple Insertions?, in: *Proceedings of the Workshop on Generative AI and Knowledge Graphs (GenAIK)*, 2025, pp. 87–99.
- [24] M. Sallam, K. Al-Salahat, H. Eid, J. Egger and B. Puladi, Human versus Artificial Intelligence: ChatGPT-4 Outperforming Bing, Bard, ChatGPT-3.5, and Humans in Clinical Chemistry Multiple-Choice Questions, *medRxiv* (2024). doi:10.1101/2024.01.08.24300995.
- [25] R. Studer, V.R. Benjamins and D. Fensel, Knowledge engineering: Principles and methods, *Data & Knowledge Engineering* **25**(1) (1998), 161–197. doi:https://doi.org/10.1016/S0169-023X(97)00056-6.
- [26] M. Trajanoska, R. Stojanov and D. Trajanov, Enhancing Knowledge Graph Construction Using Large Language Models, *arXiv preprint arXiv:2305.04676* (2023).
- [27] S. Tsaneva, G. Herwanto and M. Sabou, Benchmarking Ontology Validation Capabilities of LLMs, in: *Proceedings of the Special Session on Harmonising Generative AI and Semantic Web Technologies (HGAIS 2024) co-located with the 23rd International Semantic Web Conference (ISWC 2024), November 13, 2024, Baltimore, Maryland, CEUR-WS*, 2024.
- [28] S. Tsaneva, S. Vasic and M. Sabou, LLM-driven Ontology Evaluation: Verifying Ontology Restrictions with ChatGPT, in: *Data Quality meets Machine Learning and Knowledge Graphs, DQMLKG Workshop at ESWC 2024, CEUR-WS*, 2024.
- [29] S. Tsaneva, D. Dessì, F. Osborne, M. Sabou et al., Knowledge graph validation by integrating LLMs and human-in-the-loop, *Information Processing & Management* **62**(5) (2025), 104145. doi:https://doi.org/10.1016/j.ipm.2025.104145.
- [30] N. Tufek, A.S.S. Thuluva, T. Bandyopadhyay, V.P. Just, M. Sabou, F.J. Ekaputra and A. Hanbury, Validating Semantic Artefacts With Large Language Models, in: *The Semantic Web: ESWC 2024 Satellite Events*, Springer Nature Switzerland, Cham, 2025, pp. 92–101. doi:10.1007/978-3-031-78952-6_9.
- [31] K. Wang, G. Qi, J. Li and S. Zhai, Can Large Language Models Understand DL-Lite Ontologies? An Empirical Study, *arXiv preprint arXiv:2406.17532* (2024).
- [32] L. Wang, W. Xu, Y. Lan, Z. Hu, Y. Lan, R.K.-W. Lee and E.-P. Lim, Plan-and-solve prompting: Improving zero-shot chain-of-thought reasoning by large language models, *arXiv preprint arXiv:2305.04091* (2023).
- [33] F. Xu, Q. Lin, J. Han, T. Zhao, J. Liu and E. Cambria, Are Large Language Models Really Good Logical Reasoners? A Comprehensive Evaluation and Beyond, *IEEE Transactions on Knowledge and Data Engineering* **37**(4) (2025), 1620–1634. doi:10.1109/TKDE.2025.3536008.
- [34] X. Xu, C. Tao, T. Shen, C. Xu, H. Xu, G. Long, J.-G. Lou and S. Ma, Re-reading improves reasoning in large language models, in: *Proceedings of the 2024 Conference on Empirical Methods in Natural Language Processing*, 2024, pp. 15549–15575.
- [35] B. Zhang, I. Reklós, N. Jain, A.M. Peñuela and E. Simperl, Using Large Language Models for Knowledge Engineering (LLMKE): A Case Study on Wikidata, *arXiv preprint arXiv:2309.08491* (2023).
- [36] B. Zhang, V. Carriero, K. Schreiberhuber, S. Tsaneva, L. Gonzalez, J. Kim and J. de Berardinis, OntoChat: A Framework for Conversational Ontology Engineering Using Language Models, in: *The Semantic Web: ESWC 2024 Satellite Events*, Springer Nature Switzerland, Cham, 2025, pp. 102–121. doi:10.1007/978-3-031-78952-6_10.
- [37] Y. Zhu, X. Wang, J. Chen, S. Qiao, Y. Ou, Y. Yao, S. Deng, H. Chen and N. Zhang, LLMs for Knowledge Graph Construction and Reasoning: Recent Capabilities and Future Opportunities, *arXiv preprint arXiv:2305.13168* (2023).