

From GPT to Mistral: Cross-Domain Ontology Learning with NeOn-GPT

Nadeen FATHALLAH^a, Arunav DAS^b, Stefano DE GIORGIS^c,
Andrea POLTRONIERI^d, Peter HAASE^e, Liubov KOVRIGUINA^e,
Albert MEROÑO-PEÑUELA^b, Elena SIMPERL^b, Steffen STAAB^{a,h}, and
Alsayed ALGERGAWY^{f,g}

^a*Department of Analytic Computing, Institute for Artificial Intelligence, University of Stuttgart, Germany*

^b*King's College London, UK*

^c*Institute of Cognitive Sciences and Technologies, National Research Council, Catania, Italy*

^d*Department of Computer Science and Engineering, University of Bologna, Italy*
^e*metaphacts GmbH, Walldorf, Germany*

^f*Data and Knowledge Engineering, University of Passau, Passau, Germany*

^g*Institute for Informatics, Friedrich-Schiller-University Jena, Jena, Germany*

^h*University of Southampton, Southampton, UK*

Abstract. We present the extended NeOn-GPT pipeline, an LLM-powered ontology learning pipeline grounded in the NeOn methodology. NeOn-GPT is a domain-agnostic ontology learning pipeline that comprises two components: (i) *ontology draft generation: a multi-step prompting pipeline following the NeOn methodology, including requirement specification, Competency Questions generation, ontology conceptualization and implementation, formal modelling, population, documentation*, (ii) *automated ontology verification and resolution achieved through orchestrated calls to third-party tools complemented by LLM-suggested repairs*. The extended pipeline incorporates an explicit step for reusing existing relevant domain ontologies to guide LLMs toward more consistent modeling decisions. We evaluate NeOn-GPT across four distinct domains (Wine, Cheminformatics, Environmental Microbiology, and Sewer Networks) using both proprietary (GPT-4o) and open-source (Mistral, Llama-4, DeepSeek) models. Gold-standard alignment is assessed using three complementary metrics: structural metrics (class, property, and axiom profiles), lexical metrics (exact matches and Jaro-Winkler similarity ≥ 0.8), and semantic metrics based on sentence-transformer embeddings. Results show that LLMs consistently generate ontologies with rich relational structures (including functional, transitive, and domain-range constraints) and meaningful semantic alignment, with most entity and triple similarities falling in the 0.5-0.8 range. Overall, this study provides a comprehensive, cross-domain evaluation of a NeOn-guided LLM ontology learning pipeline, clarifying its capabilities and limitations.

Keywords. Large Language Models, NeOn Methodology, Ontology Learning, Prompt Engineering.

1. Introduction

Ontologies are formal and explicit specifications of shared conceptualizations, structured representations of the entities, properties, and rules that define a domain [1,2]. They provide a semantic foundation for data integration, interoperability, and reasoning in domains such as biomedicine, environmental modelling, and intelligent decision support [3,4,5,6]. [The construction of high-quality ontologies, however, remains labour-intensive, requiring domain expertise, methodological rigour, and significant manual effort.](#)

Recent advances in large language models (LLMs) offer a promising opportunity to alleviate this bottleneck. LLMs are increasingly used to transform natural language descriptions into structured representations, raising the question of whether they can support or even automate ontology-engineering pipelines. This challenge is both practically important and scientifically significant: a reliable LLM-assisted pipeline would reduce development time, broaden accessibility, and strengthen ontology reuse across domains.

[However, automating ontology engineering with LLMs remains difficult. Naïve prompting strategies often produce incomplete conceptual models, missing axioms, or logically inconsistent structures. Moreover, LLMs struggle to reuse existing ontologies, which results in hallucinated entities or incorrect reuse of vocabulary. These limitations make end-to-end automation challenging, despite the impressive generative capabilities of current models.](#)

Existing approaches have attempted to leverage LLMs for isolated ontology learning tasks, such as entity extraction, triple generation, and ontology validation [7,8,9,10,11,12,13,14,15], but they do not provide a methodology-driven, end-to-end ontology construction process with robust verification and resolution. [In particular, existing LLM-based methods typically overlook systematic syntax checking, logical consistency verification, and modelling-quality assessment, resulting in ontologies that are often invalid or unusable in downstream reasoning tasks.](#)

Our earlier work, NeOn-GPT [16,17], introduced a prompt-engineered ontology construction pipeline grounded in the NeOn methodology and coupled with a verification-and-resolution stage that integrates third-party tools (RDFLib, HermiT, Pellet, OOPS!) with LLM-guided repairs. This combination enabled the automatic detection and correction of syntax errors, logical inconsistencies, and modelling pitfalls, producing draft ontologies that were consistently valid, unlike the outputs of prior LLM-based approaches [18].

The contributions of the original NeOn-GPT [16] paper are:

- Implementation of a prompt pipeline grounded in the NeOn methodology.
- NeOn-GPT, an LLM-driven ontology generation pipeline including multi-step ontology draft generation phase and an extensive validation phase encompassing syntax, consistency, and modelling pitfalls check of the generated ontology.
- An empirical evaluation of NeOn-GPT using a single gold-standard ontology, the Stanford Wine Ontology, and a single LLM, GPT-3.5, comparing zero-shot prompting, a prompt pipeline derived from the Ontology Development 101 methodology [19], and the NeOn-GPT pipeline across structural metrics, hierarchy depth, axioms, properties, individuals, and inference behavior.

Our work [16,17] showed that LLM-generated ontologies differ from expert-curated gold standard ontologies, particularly in the depth of hierarchies and the coverage of domain-specific concepts. Expert-curated ontologies outperform LLM-generated ones not only because they are grounded in expert knowledge, but also because ontology engineers reuse existing domain knowledge, both ontological and non-ontological resources, during construction.

Moreover, our prior evaluation was limited to the Wine ontology, a domain likely well represented in LLM training data. To meaningfully assess LLM capabilities, a broader evaluation is required across domains with varying degrees of representation, including sparsely modelled domains.

This paper presents an enhanced version of NeOn-GPT¹. We integrate a reuse stage of ontological and non-ontological resources in accordance with the NeOn methodology guidelines. We broaden the evaluation to assess LLM capability across heterogeneous domains, ranging from well-established ontologies to sparsely modelled knowledge areas.

This paper makes the following contributions:

- We incorporate the reuse stage into the NeOn-GPT pipeline, allowing the identification, evaluation, and incorporation of curated ontology fragments during conceptual modelling (Section 3.1.4).
- We expand the empirical evaluation to four heterogeneous domains (wine, sewer networks infrastructure, cheminformatics, and environmental microbiology), demonstrating the generality of the approach (Sections 5 and 6).
- We compare multiple LLMs, including GPT-4o and open-source models such as Mistral, LLama, and Deepseek to assess model dependence and robustness (Section 5).
- We develop and apply a comprehensive evaluation framework for LLM-generated ontologies that assesses structural, lexical, and semantic alignment with gold-standard ontologies, enabling a multi-dimensional analysis of ontology quality beyond surface-level checks (Section 4).

The paper is structured as follows: Section 2 reviews related literature. Section 3 presents our methodology. Section 4 outlines the evaluation metrics. Section 5 describes the experimental setup and 6 presents results and discussions. Section 8 discusses the limitations of our approach. Finally, section 9 concludes with a summary of contributions and presents directions for future work to address the limitations.

2. Related Work

2.1. Ontology Learning from Text

Ontology learning from text refers to the automatic or semi-automatic process of acquiring ontologies from unstructured textual data to construct them. This task has been a long-standing goal in the Semantic Web community, aiming to reduce the cost and com-

¹The supplementary materials, including code, prompting templates, evaluation scripts, and generated ontologies, are available at: <https://github.com/NadeenAhmad/neon-gpt-extended>

plexity of manual ontology engineering while improving scalability and domain coverage [20,21]. Early approaches used rule-based and statistical methods; a prominent example is Hearst patterns, a rule-based method that identifies taxonomic relations such as hyponymy—the relation between a general entity and its more specific instances (e.g., "animal" and "dog")—using lexico-syntactic cues, i.e., fixed linguistic patterns like "X is a type of Y" that signal hierarchical relationships in text [22]. [23] proposed a semi-automatic approach for constructing ontologies from domain-specific corpora by identifying relevant terms and extracting semantic relations such as synonymy, hypernymy, and meronymy through linguistic and contextual analysis. [24] proposed a semi-automatic ontology learning framework that integrates natural language processing (NLP) to extract ontological structures from unstructured and semi-structured web data. Their approach involves multiple stages, including ontology import, entity and relation extraction, pruning, refinement, and evaluation, with an emphasis on human-in-the-loop guidance to enhance accuracy and domain relevance. The Text2Onto framework [25] introduces a probabilistic ontology model, which assigns confidence values to extracted elements such as concepts, taxonomic relations, and non-taxonomic relations. The framework supports incremental updates and user feedback, enabling continuous refinement of ontologies and enhanced uncertainty management in learned structures. [26] reviewed the transition from shallow learning techniques—such as association rule mining and clustering—to deep learning architectures capable of capturing richer semantic patterns. More recent efforts leverage neural networks and pre-trained language models to extract ontological components from large corpora with minimal supervision [27].

2.2. *Ontology Engineering Methodologies*

Ontology engineering methodologies provide structured processes and best practices for developing, maintaining, and reusing ontologies. These methodologies guide developers through stages such as requirements specification, conceptualization, formalization, evaluation, and deployment, aiming to ensure both the semantic quality and practical utility of the resulting ontologies [28,29].

One of the earliest and most widely adopted methodologies is METHONTOLOGY, which defines a waterfall-like process encompassing specification, conceptualization, integration, implementation, and maintenance. It also emphasizes supporting activities such as knowledge acquisition and evaluation [29]. [30] proposed DILIGENT, a collaborative ontology development in distributed environments, offering mechanisms for managing user feedback and reconciling conflicting edits. SAMOD (Simplified Agile Methodology for Ontology Development) [31] is an agile methodology, that promotes ontology development through small, iterative steps. It emphasizes the use of exemplar domain descriptions and real-world data to create ontologies. This approach facilitates the early detection of inconsistencies, ensuring that the ontology remains aligned with practical use cases. RapidOWL [32] is another agile methodology for collaborative and iterative development of ontologies. RapidOWL's agile paradigm relies on iterative refinement, annotation, and structuring of a knowledge base, allowing for the selective addition, removal, or annotation of information chunks. This methodology is suitable for scenarios that require lightweight, easy-to-implement solutions to support spatially distributed and highly collaborative environments. eXtreme Design (XD)[33] is a collaborative, iterative methodology that emphasizes the use of Ontology Design Patterns

to address recurring modeling problems. XD adopts practices from agile software engineering, such as test-driven development, pair programming, and modularization, and applies them to ontology engineering. The methodology guides developers through identifying relevant ODPs, adapting them to specific use cases, and integrating them into the ontology, thereby promoting reusability and consistency across ontology projects.

The NeOn methodology defines ontology development through a set of concrete activities and artefacts organised into nine development scenarios [34,35]. A scenario in NeOn represents a typical project situation, for example, building an ontology from scratch, reusing existing ontologies, reengineering non-ontological resources (e.g., XML schemas or databases), or integrating several ontologies into a network. Each scenario specifies the operational steps involved, such as drafting an Ontology Requirements Specification Document, identifying and evaluating candidate ontologies for reuse, extracting or restructuring modules, aligning heterogeneous resources, documenting modelling decisions, and performing evaluation. Rather than enforcing a fixed sequence, NeOn allows these steps to be combined and repeated as needed, enabling both iterative–incremental and waterfall-style execution. More recently, the Linked Open Terms (LOT) methodology [36] streamlines ontology development by adopting a lightweight set of steps focused on reuse, interoperability, and Web publication. LOT inherits NeOn’s reuse orientation but simplifies the process by emphasizing minimal documentation, agile iteration, and a web-first deployment approach.

To systematically select an appropriate methodological basis for an LLM-driven ontology engineering pipeline, we conducted a comparison of methodologies using evaluation criteria adapted from prior comparative studies [37,38]. These criteria cover the following methodological capabilities:

- **Domain analysis and Requirements specification:** defines the scope of the ontology, which can prevent LLMs from drifting into unrelated topics.
- **Conceptualization:** structures domain knowledge into classes, relations, and constraints; provides a reference model that guides and constrains LLM outputs.
- **Reuse:** enables systematic incorporation of existing ontologies and vocabularies; anchors LLM generation in established resources and can reduce hallucinated terms.
- **Collaborative construction:** coordinates inputs from different contributors; helps provide guidance to the LLM and can reduce scope drift.
- **Evaluation support:** offers mechanisms to assess coherence and correctness, helping to identify inconsistencies generated by LLMs.
- **Maintenance:** manages updates and revisions across the ontology’s lifecycle; important for controlling cumulative drift across multiple rounds of LLM-assisted editing.
- **Documentation:** records the ontology’s components and their descriptions, ensuring that LLM-generated material remains understandable and usable in later development stages.
- **Integration:** supports alignment and combination of heterogeneous sources; ensures that LLM-produced fragments remain interoperable with external vocabularies and standards.

Methodology	Domain Analysis		Reuse Support	Collaborative Construction	Evaluation Support	Maintenance Support	Documentation Support	Integration / Merging
	Requirements Specification	Conceptualization						
METHONTOLOGY [29]	✓	✓	✓	✗	✓	✓	✓	✓
Ontology Development 101 [19]	✓	✓	✓	✗	✓	✗	✓	✓
DILIGENT [30]	✓	✓	✗	✓	✓	✗	✗	✗
SAMOD [31]	✓	✓	✗	✗	✓	✓	✗	✗
RapidOWL [32]	✓	✓	✗	✓	✗	✗	✗	✗
eXtreme Design (XD) [33]	✓	✓	✓	✓	✓	✗	✗	✗
LOT [36]	✓	✓	✓	✓	✓	✓	✓	✓
NeOn [34,35]	✓	✓	✓	✓	✓	✓	✓	✓

Table 1. Comparison of ontology engineering methodologies using criteria adapted from [37,38]

Our comparison in Table 1 shows that, unlike METHONTOLOGY and Ontology 101, NeOn supports collaborative ontology design by structuring how different contributors, such as domain experts and ontology engineers, supply domain descriptions, examples, and reuse resources. This aligns well with LLM-based pipelines, where coordinated inputs can help prevent LLMs from drifting in the ontology’s scope.

Collaborative methodologies such as DILIGENT focus primarily on governance and conflict resolution in distributed settings, but they do not cover reuse. Agile and lightweight approaches, including SAMOD, RapidOWL, and XD, prioritize rapid iteration and minimal documentation. While effective for small or well-bounded ontologies, they do not support documentation and integration, which is problematic in an LLM-based pipeline to safeguard against scope drift and modeling inconsistencies that may be introduced by LLMs.

NeOn and LOT provide support for all criteria in Table 1, both are therefore reasonable candidates as methodological scaffolds for LLM-assisted ontology engineering. While both methodologies offer comprehensive support, they differ in how reuse is operationalised. NeOn provides scenario-specific procedures for identifying, evaluating, adapting, and integrating existing ontologies, ontology design patterns (ODPs), and non-ontological resources, including steps for module extraction, reengineering, combining multiple ontologies, and reusing ontology modules. LOT adopts a lightweight, implementation-oriented reuse strategy in which reuse is carried out at the level of individual terms or modules from existing ontologies. It doesn’t provide guidelines for identifying and evaluating candidate ontologies, reusing non-ontological resources, or combining multiple ontologies during development. In an LLM-based setting, NeOn’s procedural treatment of reuse is advantageous because it provides constraints for grounding LLM-generated content in established resources, thereby reducing the likelihood of hallucinated terms or unsupported modeling choices. In this work, we adopt the NeOn methodology as the basis for structuring our LLM-driven ontology generation pipeline.

2.3. LLMs for Ontology Learning

The application of LLMs to ontology engineering has evolved through distinct phases, from automating discrete tasks [39,40,41,42] toward systems that aim for full ontology generation [43,44,45,46,47]. A critical analysis of this progression reveals common limitations related to methodological grounding, verification, and scalability, which our work on NeOn-GPT directly addresses.

Discrete Task Automation. Early research established the potential of LLMs as powerful extractors of ontological components. Using zero-shot or few-shot prompting, studies focused on isolated tasks such as identifying concepts, subclass relations, and

instances from text [7,8,9]. This foundational work was extended to other specific activities, including generating competency questions for existing ontologies [10,11], ontology refinement [12], population [13], and alignment [14]. While these approaches demonstrate that LLMs can effectively automate individual sub-tasks, they operate as point solutions. A significant gap remains in orchestrating these discrete capabilities into a cohesive, end-to-end pipeline for building new ontologies from scratch. Furthermore, the outputs of these isolated tasks are often generated without being integrated into a unified ontological model, making it challenging to evaluate their cumulative impact on the overall coherence and quality of the final ontology. The generated outputs are also typically not subjected to rigorous syntactic or logical validation, leaving their practical utility for formal knowledge representation uncertain. Our NeOn-GPT pipeline is designed to address this gap by incorporating dedicated phases for these core tasks, including requirement specification, competency question generation, and ontology population, within a unified pipeline, enabling the assessment of their collective contribution to the final ontology’s quality.

Ontology Generation Pipelines. A significant shift has occurred with the development of systems for full ontology generation, which aim to move beyond isolated tasks.

The first line of work examines the basic ontology-generation capabilities of LLMs under different prompting techniques. A representative example of this line of work is the study by [18], which evaluates zero-shot, one-shot, and few-shot prompting for generating capability ontologies from natural-language descriptions. The study assesses the outputs through OWL syntax checks, reasoning, and constraint validation. Their results show that few-shot prompting substantially improves structural correctness and completeness, while zero-shot prompting frequently leads to incomplete or syntactically invalid axioms. In several cases, the generated ontologies required manual correction to resolve syntax errors and logical inconsistencies before they could be used. Moreover, ontology creation is treated as a single-step generation task tied to a fixed schema: the process does not incorporate requirements analysis, conceptualization, reuse of existing ontologies, or integration stages, nor does it support extending beyond the predefined structure.

The second line of work explores advanced prompting strategies. A representative of this paradigm is Ontogenia [48], which employs an advanced prompting technique, Metacognitive Prompting, to guide the LLM through a self-reflective ontology creation process grounded in XD methodology. Its multi-stage process involves decomposing competency questions to identify entities and properties, followed by enrichment of axioms. However, its foundation in XD limits its reuse mechanism primarily to a static, pre-selected set of ODPs. This makes it less domain-agnostic, as it cannot function effectively where relevant ODPs are unavailable. In contrast, NeOn’s broader reuse guidelines can leverage ODPs when they exist, but are not dependent on them; they can alternatively incorporate ontologies or non-ontological resources. A further consequence of the XD methodology is its lack of documentation guidelines, as noted in Section 2.2, which is critical in LLM-based pipelines for traceability and mitigating scope drift. This limitation is directly evidenced in their results, where all generated ontologies consistently lack annotations, as flagged by the OOPS! Pitfall Scanner (P08). Their evaluation also reveals persistent modeling errors, such as incorrect domain/range assignments (P19) and missing inverse relationships (P13). The NeOn-GPT pipeline addresses these issues: it includes a dedicated formal modeling phase that prompts for property characteristics,

such as inverses, and its integrated verification loop utilizes OOPS! to detect and correct critical and important pitfalls, like P19, through iterative refinement.

The third line of work investigates LLM fine-tuning to internalize principles of ontology engineering. A representative of this paradigm is the work by [49], which investigates the fine-tuning of LLMs on foundational ontology engineering textbooks to internalize syntactic and conceptual knowledge. Their approach relies on a single-stage, one-shot generation process, where a fine-tuned model is prompted to produce a complete ontology in one step. While this approach enhances the model’s understanding of general ontology engineering principles, it also reveals key limitations. The one-shot generation conflates the distinct, sequential stages of ontology engineering into a single, opaque transformation, which inherently constrains the scale and complexity of the output. This is evidenced by their results, where the generated ontologies remain very small in scale compared to expert-curated benchmarks. Furthermore, the technique’s effectiveness was highly model-dependent; notably, the fine-tuned Mistral 7B struggled with syntactic precision, suggesting that smaller models did not benefit robustly from their approach. In contrast, the NeOn-GPT pipeline is a multi-stage, iterative process that decomposes ontology creation into managed phases, enabling the systematic construction of ontologies.

The fourth line of work explores agentic AI systems for ontology engineering through interactive, human-in-the-loop workflows. Recently, such an approach has been implemented in *metis*, an AI agentic platform by metaphacts [50]. Its Semantic Modeling Assistant supports collaborative ontology development by guiding users through a structured modeling process based on the Linked Open Terms (LOT) methodology and metaphacts’ Semantic Modeling Guidelines². The assistant helps formulate competency questions, supports reuse via the metaphacts Ontology Repository³, and assists with conceptualization, encoding, and evaluation through user-facing guidance, SPARQL-based checks, and LLM-based critique. While *metis* provides interactive guidance, suggestions, and validation during the modeling process, NeOn-GPT differs in that it systematically prompts the LLM to generate and enrich formal ontology axioms—including object properties, inverse properties, and other property-level axioms—and integrates external OWL reasoners and modeling-pitfall detection tools to iteratively verify and repair these generated artifacts.

The novelty of NeOn-GPT does not lie in introducing new individual ontology learning techniques, but in the methodology-driven orchestration of existing LLM capabilities—spanning requirements specification, reuse, multi-stage generation, and integrated verification and repair—into a single, end-to-end pipeline whose outcome is a directly usable, verified ontology artifact.

Finally, existing work treats syntactic validation, logical consistency checking, and modeling pitfall detection as isolated or post hoc checks, rather than integrating them into a unified ontology generation process that systematically produces verified and repaired ontology artifacts. Furthermore, the impact of applying a single methodology-guided pipeline across ontology domains with substantially different structures and complexities has not been systematically investigated. In this work, we address these gaps by applying the NeOn-GPT pipeline to four heterogeneous domains: wine, sewer networks infrastructure, cheminformatics, and environmental microbiology.

²https://metaphacts.com/images/PDFs/metaphacts_Semantic_Modeling_Guidelines_2.0.pdf

³<https://ontologies.metaphacts.com/>

2.4. Prompt Engineering for Knowledge Engineering Tasks

LLMs exhibit significant variability in output quality depending on how prompts are formulated. Subtle changes in wording, structure, or context can lead to different responses. As a result, prompt engineering has emerged as a critical practice for guiding LLMs toward producing accurate, coherent, and context-aware outputs, particularly in complex tasks such as ontology engineering [51,52]. Prompt engineering involves systematically designing instructions and examples that shape the model’s reasoning process and output format to align with task-specific requirements.

The remainder of this subsection reviews the four prompting techniques adopted in NeOn-GPT: few-shot prompting, role-play prompting, chain-of-thought prompting, and iterative prompting with feedback.

Few-shot prompting introduces several representative input-output examples within the prompt to illustrate the task. This approach enables LLMs to generalize task structure without explicit fine-tuning [53,54]. It is widely used in knowledge extraction and triple generation tasks and has been shown to improve consistency in entity and relation extraction [55].

Role-play prompting instructs the model to adopt a specific persona or domain role (e.g., “You are an ontology engineer”). This method leverages the LLM’s contextual alignment capabilities to shape its behavior and terminology [56,57]. It is particularly useful for ontology modeling tasks that require adherence to formal constraints and expert-level language.

Chain-of-thought prompting encourages the LLM to generate intermediate reasoning steps before producing the final output [58,59]. This enhances performance on tasks that require logical inference or sequential reasoning.

Iterative prompting with feedback is a technique where the LLM is prompted to revise its own output based on diagnostic messages or evaluation results [60]. In question answering pipelines, iterative prompting has been used to improve factual accuracy by feeding the model’s initial answer into a verification module (e.g., fact-checker or retrieval system), then prompting the LLM to revise its response based on detected inaccuracies or missing context [61].

3. Methodology

In this section, we present the NeOn-GPT methodology, which is organised into two parts, as illustrated in Figure 1:

- The first part, *ontology draft generation* (Sections 3.1–3.1.7, Figure 2: Steps 1-7), operationalizes NeOn’s activities for requirements specification, conceptualization, reuse, implementation, formal modelling, and population to produce a Turtle ontology draft. In this extended version, these stages differ from the original NeOn-GPT pipeline through the inclusion of an explicit NeOn-guided reuse stage, as described in Section 3.1.4 (Figure 2: Step 4 in green).
- The second part, *ontology verification and resolution* (Section 3.2), relies on third-party tools to perform automated verification, with the LLM used to iteratively repair any issues surfaced during verification.

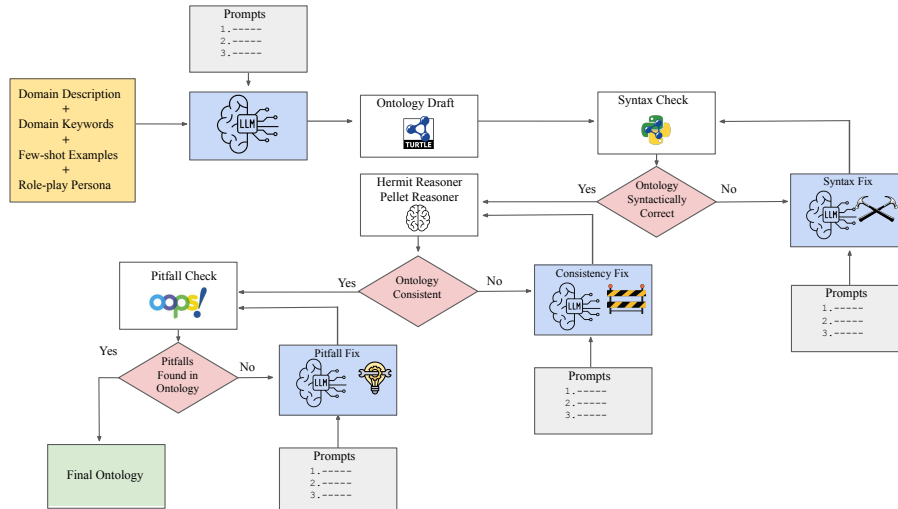


Figure 1. NeOn-GPT pipeline for ontology learning and verification. The pipeline consists of two components: (i) *ontology draft generation* (Section 3.1) and (ii) *ontology verification and resolution* (Section 3.2). Domain-specific inputs (domain description, domain keywords, few-shot examples, and role-play personas) prompt the LLM to generate an ontology draft in Turtle syntax. The draft then undergoes a three-stage verification and resolution process: syntax checking using RDFLib, logical consistency checking via OWL reasoners (Hermit and Pellet), and pitfall detection using the OOPS! service. Identified issues trigger corrective re-prompting cycles to fix syntax errors, logical inconsistencies, and common modeling pitfalls. This figure is adapted from our original NeOn-GPT architecture introduced in [16,17].

Our original NeOn-GPT pipeline [16] performed well in domains such as Wine, where a compact, well-structured reference ontology is available, but it struggled in sparsely modelled domains, most notably in the life sciences [17]. In such settings, the LLM often lacked sufficient grounding to generate coherent conceptual models, resulting in shallow hierarchies, inconsistent terminology, and missing domain-specific relationships. These limitations motivated the introduction of the reuse stage in this extended methodology (Section 3.1.4), in which curated fragments of expert-maintained ontologies are incorporated in accordance with NeOn’s reuse guidelines. This addition strengthens the conceptual model and improves the reliability of the generated ontology draft in domains with limited lexical or structural coverage.

3.1. Ontology Draft Generation

The ontology draft generation part takes as input the natural language domain description (120-150 words), the curated few-shot examples, the role-play persona, and 10-15 domain-specific keywords, shown in Figure 2. These inputs initiate the stepwise execution of the NeOn-aligned stages: requirement specification, competency-question generation, conceptual modeling, reuse, implementation, formal modeling, and enrichment & population. The output of this part is an ontology draft in Turtle syntax, including classes, properties, axioms, and named individuals.

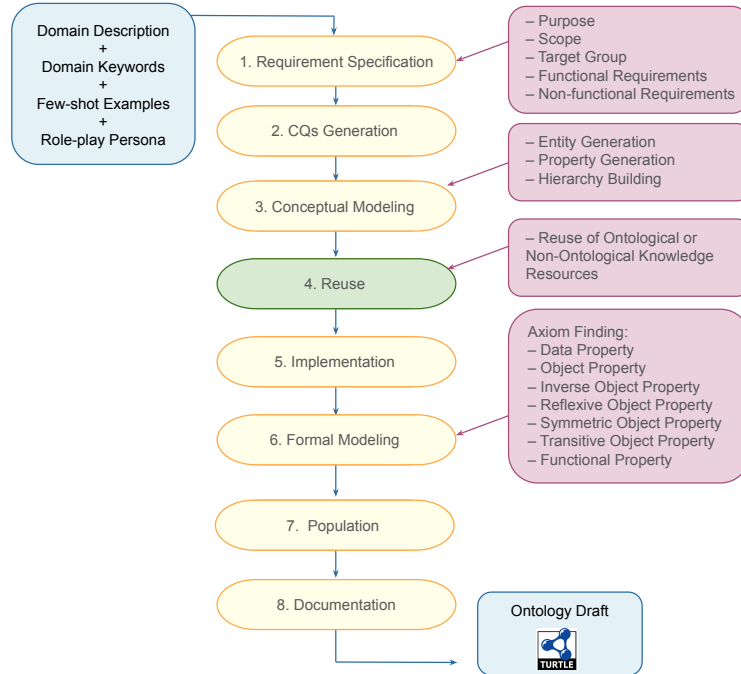


Figure 2. Overview of the NeOn-GPT ontology draft generation stages. The process begins with the specification of requirements, followed by the generation of competency questions (CQs), conceptual modeling, and the reuse of existing knowledge resources. The pipeline proceeds through implementation, formal modeling (axiom enrichment), and documentation, and concludes with ontology enrichment and population, which results in an ontology draft serialized in Turtle syntax.

3.1.1. Specification of Ontology Requirements

As shown in **Figure 2: Step 1**, ontology draft generation begins with the specification of requirements. In the NeOn methodology, this activity defines the purpose, scope, target users, intended uses, and functional and non-functional requirements of the ontology. To initiate this stage, the LLM is provided with a natural-language description of the domain (120-150 words) together with a short list of 10-15 domain-relevant keywords (as shown in Figure 2), which supply the contextual grounding needed for requirement elicitation. We operationalize this stage using a two-layer prompting strategy: *role-play prompting* and *chain-of-thought prompting*.

Role-play prompting. Building on empirical findings from our prior work [57,17], which show that prompting LLMs with domain-specific personas increases factual accuracy in LLM responses. We employ *domain-specific personas* to provide the LLM with an appropriate epistemic and semantic grounding. These personas are generated using GPT-4o based on the prompt template shown in Appendix A.1. Each persona defines a domain expert with relevant knowledge, responsibilities, and modeling constraints (e.g., adherence to ontology engineering principles). The automatically generated personas are then manually refined to ensure factual accuracy. This expert-like role-play primes the LLM to reason with domain-appropriate terminology and modelling expectations.

Chain-of-thought prompting. Once the persona is established, we guide the model through the NeOn requirement specification process using a chain-of-thought prompting approach. The model is instructed to articulate, in order:

- (a) the purpose of the ontology,
- (b) the scope and coverage,
- (c) the target users,
- (d) intended uses,
- (e) functional requirements, and
- (f) non-functional requirements.

This produces the requirement specifications that serve as the foundation for all subsequent steps in ontology draft generation. [The prompt template is shown in Appendix A.2: Listing 1.](#)

3.1.2. Competency Questions Generation

Following the requirement specification, we prompt the LLM to generate a set of Competency Questions (CQs) based on the defined requirements in **Step 1**. CQs are natural-language questions that define the domain knowledge and modelling requirements an ontology must support. Using **few-shot prompting**, the model is shown six example CQs and asked to produce similar questions that reflect the key domain concepts and relationships to be captured in the ontology.

To support domain relevance, the prompt also includes a curated list of 10–15 domain-specific keywords (e.g., *Atom*, *Chemical Bond*, *Molecular Entity*). These keywords serve as semantic anchors, encouraging the model to generate CQs centered around the concepts that are important in the target domain.

As shown in [Figure 2: Step 2](#), this stage takes the requirement specifications as input and produces a structured set of CQs that serves as the starting point for conceptual modelling. [The prompt template is shown in Appendix A.2: Listing 2.](#)

A few-shot CQ example used in the cheminformatics domain is:

```
"cq1": "Which bioassay is used to test the activity of Ibuprofen compound?"
```

3.1.3. Ontology Conceptualization

According to the NeOn methodology, the next stage involves ontology conceptualization, where entities and properties (or relations) are extracted and structured into a coherent model that reflects the domain's semantics ([Figure 2: Step 3](#)).

In our pipeline, this is operationalized by prompting the LLM to translate the CQs generated in **Step 2** into the subject-predicate-object (SPO) triples that represent the core concepts and properties within the domain. [This step is decomposed into multiple tasks:](#)

- **Domain entity extraction:** The model is first prompted to identify the relevant entities appearing in the CQs.
- **Hierarchy identification:** Next, the model generates subclass hierarchies that organize these entities into a coherent class structure.

- **SPO triple generation:** Finally, the model generates subject–predicate–object triples that form the core of the conceptual model.

The prompt templates are shown in Appendix A.2: Listings 3, 3.1, 3.2

Entity Generation: Using **few-shot prompting** with six annotated examples, the LLM is instructed to extract domain-relevant entities from the generated CQs. A few-shot example from the cheminformatics domain is:

```
"cq1": "Which bioassay is used to test the activity of Ibuprofen compound?"  
Entities: Bioassay, Compound
```

Properties (Relations) Generation: We apply **few-shot prompting**, providing the model with six annotated examples from the target domain that demonstrate how to identify and structure subject–predicate–object triples based on the CQs generated in the previous step. In the cheminformatics domain, for example, the few-shot examples include:

```
"cq1": "Which bioassay is used to test the activity of Ibuprofen compound?"  
Relations: Compound-testedBy-Bioassay  
           Compound-inhibitsTarget-Protein
```

Hierarchy Building: Our empirical evaluations in [17,16] show that LLM-generated ontologies exhibit limited hierarchical depth, typically only one to two subclass levels, and frequently omit the intermediate conceptual layers found in expert-curated ontologies. To mitigate this reduced structural depth, we provide the model with six few-shot examples and prompt it to organise the extracted entities into subclass hierarchies derived from the CQs. A **few-shot** hierarchy example used in the prompt is:

```
"cq1": "Which bioassay is used to test the activity of Ibuprofen compound?"  
Hierarchies: Ibuprofen-subClassOf-Compound
```

The output of this step is a conceptual model expressed as a set of subject–predicate–object triples that define the core structure of the ontology.

3.1.4. *Ontology Reuse*

The NeOn methodology emphasises ontology reuse, incorporating existing knowledge resources (e.g., domain ontologies, vocabularies) or non-ontological sources (e.g., thesauri, taxonomies, structured datasets) to support the construction of new ontologies. NeOn prescribes a sequence of reuse activities: (i) identifying candidate ontologies and other knowledge resources, (ii) evaluating them with respect to coverage, quality, and licence constraints, (iii) selecting appropriate resources or modules, and (iv) extracting and adapting relevant fragments.

Our empirical analysis revealed that supplying large ontology fragments directly to the LLM prompt for reuse reduced the model’s ability to follow the instructions, producing irrelevant or inconsistent output. Relying on the model’s internal knowledge base for reuse led to hallucinated or nonexistent classes and relations. These limitations motivated the inclusion of a reuse stage in the pipeline, enabling the controlled incorporation of curated ontology fragments [16,17].

In this extended version of the pipeline, we incorporate a dedicated NeOn-guided reuse stage (Figure 2: Step 4 in green). Ontology engineers identify, evaluate, and extract suitable fragments from expert-curated ontologies, which are then injected as few-shot examples into the prompt. This ensures that reuse is performed systematically and that the incorporated fragments provide reliable, domain-accurate guidance during ontology draft generation. Ontology engineers draw on broader domain ontologies to extract elements for reuse in accordance with NeOn’s reuse activities. These include:

- **Domain-specific ontology fragments** that supply grounded examples of relevant classes, subclass relations, and object properties for reuse in the generated ontology.
- **Structural metadata** (e.g., expected numbers of classes, typical subclass depth, and relation density) extracted from domain-relevant ontologies. These metrics are reused as design targets in the prompt to calibrate the size and hierarchical complexity of the generated ontology, and embedded into the prompt (e.g., "Generate at least 30 entities with three levels of subclassing").

For instance, in the cheminformatics domain, ontology engineers extract fragments from the Environmental Ontology (ENVO) to introduce environmental and biological concepts. An example snippet is:

```
:Wetland rdf:type owl:Class ;
        rdfs:subClassOf :Habitat ;
        rdfs:label "Wetland"@en ;
        rdfs:comment "A habitat type characterized by saturated soil
        conditions and the presence of water-dependent vegetation."@en .

:hasSalinity rdf:type owl:ObjectProperty ;
            rdfs:domain :WaterBody ;
            rdfs:range :SalinityLevel ;
            rdfs:label "has salinity"@en ;
            rdfs:comment "Relates a water body to the salinity level
            that characterizes it."@en .
```

The prompt first introduces the concept of reuse to the LLM, explaining that selected ontology fragments and structural metadata will be incorporated into the conceptual model. The *reuse* materials are provided as few-shot examples, followed by the previously generated conceptual model in **Step 3**. The LLM is instructed to extend this model by integrating the reused fragments where appropriate. The output of this step is an extended conceptual model that incorporates the selected reuse material. [The prompt template is shown in Appendix A.2: Listing 4.](#)

3.1.5. *Ontology Implementation*

Following the NeOn methodology, once the conceptual model has been established, the next step involves implementing the ontology in a formal representation language (**Figure 2: Step 5**). We prompt the LLM to serialize the conceptual model generated in **Step 4** into OWL using Turtle syntax. Our choice of Turtle is informed by earlier experiments in which OWL/XML and RDF/XML led to frequent syntactical errors in the LLM-generated output [16].

Implementation prompts include cautionary guidance to avoid syntax errors, logical inconsistencies, and common modeling pitfalls (e.g., Wrong inverse relationships, Cycles in a class hierarchy, Multiple domains or ranges, and Wrong transitive relationships). They also instruct the model to enclose its output between predefined delimiters (e.g., `###start_turtle###` and `###end_turtle###`) so that the generated Turtle code can be automatically extracted and saved into `.ttl` file. [The prompt template is shown in Appendix A.2: Listing 5.](#)

3.1.6. *Ontology Formal Modeling*

As shown in **Figure 2: Step 6**, formal modeling is the penultimate stage of ontology construction. In this step, the ontology is enriched with logical axioms that capture domain constraints and support reasoning. We break formal modeling into the following subtasks:

- (a) **Introducing datatype properties** to represent quantitative or literal attributes of entities.
- (b) **Introducing object-properties** (e.g., inverse, functional, symmetric, reflexive, transitive) to refine the semantics of relations.

For each subtask, the LLM is prompted to analyse the ontology draft generated in **Step 5** and propose appropriate axioms. The LLM is prompted with cautionary guidance to avoid syntax errors, logical inconsistencies, and common modeling pitfalls. The LLM is also prompted to generate axioms between predefined delimiters (e.g., `###start_turtle### ... ###end_turtle###`) to enable extraction and integration into the existing `.ttl` file.

(a) Datatype Properties. The LLM is prompted to identify opportunities to introduce `owl:DatatypeProperty` definitions with suitable domains and ranges. **Few-shot prompting** is used to illustrate how datatype properties should be modelled. For example, in the cheminformatics domain, the prompt includes:

```
:hasMolecularWeight rdf:type owl:DatatypeProperty ;
  rdfs:domain :Compound ;
  rdfs:range xsd:float ;
  rdfs:label "has molecular weight"@en ;
  rdfs:comment "Specifies the molecular weight of a chemical
  compound in Daltons."@en .
```

(b) Object Properties. We first prompt the LLM to introduce object properties that capture relationships between classes. We then prompt the LLM to further enrich the ontology by introducing the following types of object-property axioms, each handled in a separate prompting step:

- **Inverse properties**
- **Reflexive properties**
- **Symmetric properties**
- **Functional properties**
- **Transitive properties**

For each axiom type, the LLM is instructed with the axiom type and a usage example. The model is then instructed to analyze the ontology draft and to add an axiom of that type, together with a suitable domain and range, only when semantically appropriate. For example, the prompt for inverse object properties explicitly states:

“An inverse property expresses a two-way relationship between two concepts. If the ontology contains a property `hasPart`, its inverse would be `isPartOf`.”

This instruction follows the prompt template provided in Appendix A.2, Prompt 6.2. Below is an LLM-generated example in the Cheminformatics domain:

Example where the LLM generates an inverse property for an existing object property in the ontology draft

```
:isActiveIngredientOf rdf:type owl:ObjectProperty ;
  rdfs:domain :ActiveSubstance ;
  rdfs:range :DrugProduct .

:hasActiveIngredient rdf:type owl:ObjectProperty ;
  owl:inverseOf :isActiveIngredientOf ;
  rdfs:domain :DrugProduct ;
  rdfs:range :ActiveSubstance .
```

To ensure format compliance, the prompt provides guidance to prevent both syntactic and modelling errors, and instructs the LLM to place all generated properties between predefined markers (e.g., `###start_turtle###` ... `###end_turtle###`). This ensures that the new content can be cleanly extracted and inserted into the existing .ttl file. The prompt templates are shown in Appendix A.2: Listings 6, 6.1, and 6.2.

3.1.7. Ontology Population

In this stage of the draft generation part, the LLM is prompted to analyze the ontology draft and populate it with **individuals** (i.e., named instances) that instantiate the classes and properties defined in the earlier steps (**Figure 2: Step 7**), to ground the ontology in real-world data and facilitate knowledge discovery (e.g., querying and reasoning).

We use **few-shot prompting** to guide the model, providing six examples that illustrate how to represent individuals with appropriate type declarations, labels, and naming conventions. These examples help constrain the model to generate domain-appropriate, non-hallucinated instances.

To ensure correct integration into the ontology, the prompt includes cautionary instructions to avoid syntax and modelling errors, and requires the LLM to output individuals between predefined delimiters (e.g., `###start_turtle###` ... `###end_turtle###`), facilitating clean extraction into the existing .ttl file. [The prompt template is shown in Appendix A.2: Listing 7](#). Here's one of the few-shot examples used in the prompt:

```
:Acetaminophen rdf:type :Compound, owl:NamedIndividual ;
  rdfs:label "Acetaminophen"@en ;
  rdfs:comment "A commonly used analgesic and antipyretic compound, also known as paracetamol."@en ;
  :hasMolecularWeight "151.16"^^xsd:float ;
  :isTestedIn :Bioassay .
```

3.1.8. *Ontology Documentation*

In the final stage of the draft-generation part (Figure 2: Step 8), the LLM is instructed to analyze the ontology draft and enhance it with documentation and metadata (e.g., ontology IRI, ontology label, version information) that improve interpretability and reuse. The model is prompted to add `rdfs:comment` annotations for classes and properties, providing human-readable descriptions that support ontology understanding and maintenance.

Additionally, the LLM is prompted to generate standard metadata elements such as labels, IRIs, and versioning information. The output of this step is appended to the existing `.ttl` file, completing the ontology draft before it proceeds to verification and resolution. [The prompt template is shown in Appendix A.2: Listing 8.](#)

3.2. *Ontology Verification and Resolution*

The second part of the methodology takes the ontology draft as input and performs verification and resolution using the third-party tools outlined in Figure 1. In this stage, the draft undergoes multi-layered verification to identify errors related to syntax, logical consistency, and modelling quality. We adopt an iterative prompting-with-feedback strategy, in which diagnostic messages produced by these tools are incorporated into LLM prompts to guide the correction of errors.

Syntax Verification. We parse the ontology draft using the RDFLib Python library [62] to validate the Turtle serialization. If parsing fails, RDFLib returns a diagnostic message that typically includes the line number and cause of the error. We extract the corresponding fragment from the ontology and provide the LLM with (i) the RDFLib diagnostic message and (ii) the extracted fragment. The LLM is prompted to propose a corrected Turtle fragment, which replaces the original fragment in the ontology. This diagnosis-repair cycle is repeated until the ontology is successfully parsed or a maximum number of repair attempts (25) is reached.

Logical Consistency Checking. To detect and resolve logical inconsistencies (e.g., unsatisfiable classes, conflicting axioms, incompatible domain-range constraints), we integrate the Hermit [63] and Pellet [64] reasoners together with the ROBOT toolkit.

We use ROBOT as an *explanation engine* for logical inconsistency detected by the Hermit and Pellet reasoners. When ROBOT reports an inconsistency or unsatisfiable class, we invoke its `explain` functionality to generate formal justifications identifying the axioms and entities involved. From these justifications, we extract the relevant ontology fragment by deriving a subgraph of the original ontology that contains all triples involving the implicated IRIs, expanded to a fixed-depth neighborhood in the ontology graph to retain sufficient local context. This fragment, together with the ROBOT explanation text, is provided to the LLM as structured diagnostic evidence.

The LLM is prompted to propose a repair (add, delete, or replacement). The proposed patch is applied to the ontology, after which the Hermit and Pellet reasoners are re-run to assess whether the inconsistency has been resolved. This diagnosis-repair cycle repeats until Hermit reports no remaining inconsistencies or a maximum of 25 iterations is reached.

Pitfall Detection. For higher-level modelling quality, we use the OOPS! (Ontology Pitfall Scanner) service [65,66] to detect structural and modelling pitfalls, such as circular subclassing, missing disjointness, or the misuse of non-OWL relations. OOPS!

classifies each pitfall as *Critical*, *Important*, or *Minor*. Our empirical findings [16] show that LLMs reliably correct *Critical* pitfalls, but often struggle with *Important* and *Minor* ones. Accordingly, we apply LLM-guided repair only to pitfalls in the *Critical* category.

When OOPS! reports a pitfall that points to specific erroneous axioms, such as P03 (“Creating the relationship ‘is’ instead of using `rdfs:subClassOf`, `rdf:type`, or `owl:sameAs`”), we extract the relevant ontology fragment by deriving a subgraph of the ontology centered on those erroneous axioms. The LLM is then prompted with (i) the OOPS! diagnostic message and (ii) the extracted problematic fragment(s). The LLM is prompted to propose a repair (add, delete, or replacement), and the proposed patch is applied to the ontology.

For pitfalls that do not correspond to specific axioms, such as P10 “Missing disjointness,” where the ontology lacks any `owl:disjointWith` assertions, the prompt instead includes (i) the OOPS! diagnostic message, and (ii) the full ontology. The LLM is prompted to propose a repair (add, delete, or replacement), and the proposed patch is applied to the ontology. This diagnosis-repair cycle repeats until OOPS! reports no remaining critical pitfalls or a maximum of 25 iterations is reached.

Finally, once all syntax, logical consistency, and critical pitfall repairs have been applied, the ontology is subjected to a final verification pass in which syntax validation, logical reasoning, and critical pitfall detection are executed again. This final check ensures that repairs introduced at one stage do not reintroduce errors at another (e.g., logical repairs that introduce syntax errors, or pitfall corrections that introduce logical inconsistencies), and that the resulting ontology satisfies all verification criteria simultaneously. The prompt template is shown in Appendix A.3

4. Evaluation

To assess the LLM’s ability to represent internal parametric domain knowledge as formal knowledge representations, we compare the generated ontologies with gold-standard benchmarks using structural analysis, lexical similarity, and semantic similarity as outlined in Figure 3. Our analysis does not aim to measure the completeness of the generated model; rather, it uses these expert-curated references (gold-standard ontologies) to diagnose the typology of knowledge gaps inherent to models operating without domain-specific corpora. Gold-standard evaluation is widely used for assessing automatically constructed ontologies, where the generated model is compared against an expert-curated reference to quantify its structural and semantic quality [67,68,69]. However, we do not treat the gold standard as a replication target, but rather as a baseline for distinguishing between simple omissions and the fundamental representational domain knowledge deficits of parametric memory.

4.1. Evaluation Metric Selection

Our evaluation metrics follow established practices in ontology quality assessment. **Structural metrics** such as the number of classes, object and data properties, axioms, and hierarchy depth are widely used in ontology-evaluation frameworks. Gangemi et al. [71] identify quantitative structural indicators—including class counts, hierarchy

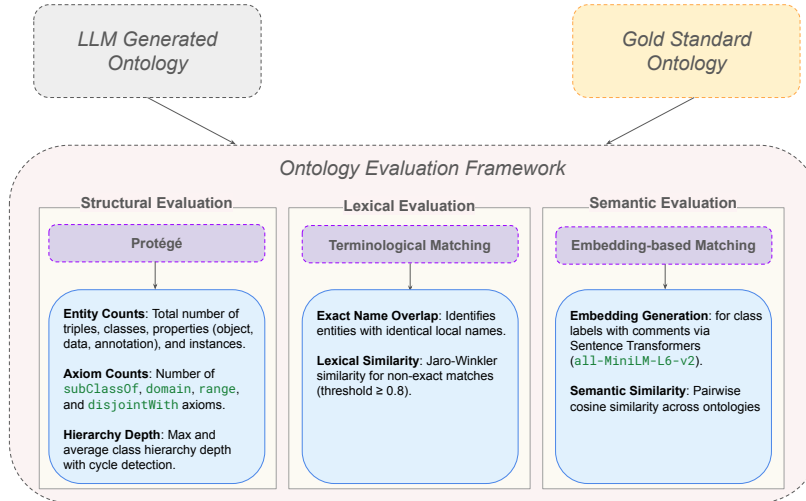


Figure 3. Overview of the ontology evaluation framework used to assess LLM-generated ontologies against expert-curated gold standards. The framework comprises three core modules: structural evaluation via Protégé [70], lexical evaluation using terminological matching, and semantic evaluation based on embedding similarity. Each component applies techniques and metrics to quantify structural fidelity, lexical similarity, and semantic similarity.

depth, and the number of taxonomic relations—as core measures for assessing ontology completeness and taxonomic organization. Fernández et al. [72] provide empirical evidence that variations in these quantitative features, such as the number of classes, depth of subclass hierarchies, and counts of properties and axioms, have measurable effects on ontology quality and knowledge reuse. These findings reinforce the role of count-based structural metrics as a foundational means of assessing the robustness and modeling characteristics of generated ontologies. Recent LLM-based studies similarly rely on structural indicators—such as class counts, property counts, axiom counts, and hierarchy depth—to evaluate the quality of automatically generated ontologies and knowledge graphs and to assess their adherence to domain requirements [73,15,74,75,76,49,18].

Lexical metrics measure terminological alignment between ontologies and are widely used in ontology matching and evaluation systems. String-based similarity methods have been shown to reliably detect terminological correspondences across ontologies, even when labels differ slightly. Cheatham and Hitzler [77] demonstrate the effectiveness of such metrics for identifying concept-level alignments, and lexical similarity is a central component of many ontology alignment frameworks (e.g., RiMOM [78]). Corpus-based ontology evaluation methods similarly rely on lexical comparison, measuring the overlap between ontology labels and terms extracted from domain corpora [79]. Moreover, gold-standard evaluation frameworks incorporate lexical evaluation to assess the coverage of generated ontologies with respect to reference terminology [80]. Recent studies of LLM-generated ontologies have similarly assessed the lexical alignment of generated classes and properties with expert vocabularies [74,81,49,82].

Semantic similarity metrics, computed using word embeddings for pairs of concepts derived from the gold-standard and LLM-generated ontologies and measured via

cosine similarity, are increasingly used to assess conceptual similarity beyond surface-level lexical matches. Ontology embedding approaches based on textual annotations (e.g., labels and definitions) demonstrate that such representations can preserve meaningful semantic relationships between classes [83]. Similar techniques are widely applied in ontology alignment, where cosine similarity between embedded class labels or descriptions is used to identify semantic mappings across ontologies [84,85,86,87]. Embedding-derived similarity has also been employed to evaluate mappings against gold-standard annotations, providing a more robust signal than purely lexical overlap [88]. Recent work on LLM-generated ontologies employs a similar strategy, utilizing embedding-based semantic similarity to quantify the correspondence between generated classes and relations and expert-curated models [15,89]. Together, these findings support our use of embedding-based measures to evaluate whether LLM-generated ontologies capture the intended conceptual semantics of domain ontologies.

4.2. Structural Evaluation

Structural evaluation examines the ontological structures produced during [NeOn-GPT's Ontology Draft Generation \(Steps 3–8\)](#), assessing how LLM-generated ontologies organize classes, properties, constraints, individuals, and documentation relative to expert-curated ontologies. All metrics are extracted using [Protégé's ontology metrics and property panels \[70\]](#).

- **Entity and Axiom Counts:** Total numbers of classes, properties (`rdf:Property`, `owl:ObjectProperty`, `owl:DatatypeProperty`, `owl:AnnotationProperty`) and axioms such as `owl:EquivalentClass`, `owl:disjointWith`. These metrics characterize the overall structural footprint generated during Steps 1–3 (Section 3.1.1–3.1.3).
- **Hierarchy Depth:** Maximum class-hierarchy depth computed over `rdfs:subClassOf` paths, reflecting the ability of Step 3 (Section 3.1.3) to construct multi-level hierarchies.
- **Property Characteristics:** Counts of inverse, functional, transitive, symmetric, and reflexive object-property features (Step 6 Section 3.1.6). These quantify how LLMs assign semantic behaviors to relations.
- **Domain and Range Assertions:** Number of domain and range constraints attached to generated properties (Step 6 Section 3.1.6), indicating whether relational vocabulary is accompanied by appropriate typing.
- **Individuals:** Number of instances generated in Step 7 (Section 3.1.7), assessing population behavior across domains.
- **Documentation and Annotations:** Counts of labels, comments, and metadata annotations (Step 8 Section 3.1.7), reflecting the human-readability and documentation structure of generated ontologies.

However, structural evaluation captures only the size and organization of an ontology; it does not assess whether the generated terminology aligns with domain vocabulary, motivating the complementary lexical evaluation that follows.

4.3. Lexical Evaluation

Lexical evaluation investigates the degree of terminological overlap and similarity between the LLM-generated and gold standard ontologies.

- **Exact Name Overlap:** Entities and properties whose local names appear verbatim in both ontologies are identified as exact lexical matches.
- **Lexical Similarity:** To capture near-matches where labels differ only slightly, we apply the Jaro-Winkler similarity metric [90], a well-established string-similarity measure widely used in ontology alignment [77]. We adopt a Jaro-Winkler threshold of 0.8 to identify high-confidence lexical matches, consistent with prior work in record linkage and approximate string search, where Jaro-Winkler scores ≥ 0.8 have been shown to yield high recall and precision in matching tasks [91,92,93,94].

Yet lexical similarity remains sensitive to naming variation and cannot capture semantically equivalent concepts expressed with different labels, motivating the semantic evaluation layer introduced next.

4.4. Semantic Evaluation

Semantic evaluation measures the conceptual correspondence between the LLM-generated ontology and the gold-standard ontology using dense vector embeddings and cosine similarity. It consists of two complementary components:

- **Entity-Level Semantic Similarity:** For every entity in both ontologies, a semantic representation is constructed from its textual description. Each entity is encoded as “*label [SEP] comment*” (with fallbacks to the local name when labels or comments are missing). Embeddings are generated using the SentenceTransformer model `all-MiniLM-L6-v2`. Cosine similarity is then computed between each LLM entity and *all* gold entities, and the highest-scoring gold entity is selected as its semantic match. Similarities are grouped into predefined buckets (e.g., <50 , $50-60$, \dots , $90-100$) to summarize alignment quality. The model `all-MiniLM-L6-v2` was selected because it is a widely used, computationally efficient sentence-embedding model that offers state-of-the-art performance on semantic similarity tasks relative to models of similar size [95]. Independent benchmark studies (e.g., MTEB [96]) demonstrate that MiniLM-based SentenceTransformers consistently achieve strong performance across semantic similarity, retrieval, and clustering tasks, while remaining lightweight enough to efficiently embed thousands of ontology entities and triples. This balance of semantic quality and scalability makes `all-MiniLM-L6-v2` suitable for ontology-level evaluation, where many embeddings must be computed.
- **Triple-Level Semantic Similarity:** Schema-level triples (e.g., `subclassOf`, `domain`, `range`, `disjointWith`, `rdf:type`) are embedded by composing the embeddings of their constituent elements into a textual sequence of the form “*subject [SEP] predicate [SEP] object*”. The same embedding model is applied to produce a dense representation for each triple. For every LLM-generated triple, cosine similarity is computed against all gold triples, and the highest-scoring gold triple is identified as the closest semantic counterpart. Triple similarities are then bucketed

in the same manner as entity similarities, enabling comparison of schema-level semantic alignment.

Together, these two measures capture semantic correspondence at both the entity and relational levels, allowing us to assess whether the generated ontology encodes concepts and schema patterns that are semantically compatible with those in the gold standard ontology, even when lexical forms or structural patterns differ.

5. Experiments

To evaluate the effectiveness and adaptability of the extended NeOn-GPT pipeline, we conduct experiments across four domains using two LLMs. We aim to assess how well the pipeline generalizes across knowledge domains of varying complexity and how its performance differs depending on the underlying LLM.

5.1. Domains

We select four domains that reflect a spectrum of structural depth and the varying levels of domain expertise required for ontology construction:

- **Wine:** A domain covering grape varieties, wine types, production processes, and sensory attributes. This domain requires moderate domain knowledge, as concepts are widely documented and terminology is stable across public sources.
- **Cheminformatics:** A scientific domain involving chemical compounds, bioassays, and experimental properties. Ontology development in this domain requires chemical and biochemical expertise, as well as familiarity with controlled vocabularies and standardized identifiers.
- **Environmental Microbiology:** A complex life-sciences domain modeling sub-surface ecosystems and biological interactions. Constructing ontologies in this domain requires expert-level knowledge of microbial taxonomy, ecological processes, and environmental measurement protocols.
- **Sewer Networks:** A domain concerning urban infrastructure and wastewater systems. Ontology development in this area requires applied engineering knowledge, including network components, operational processes, and physical infrastructure constraints.

5.2. Gold Standard Ontologies

The ontologies used in the assessment across all domains are as follows:

- **Wine domain:** Several benchmark ontologies are available for the wine domain, including the Stanford Wine Ontology and domain-specific extensions such as the EC-BACO ontology. In our evaluation, we select the widely used *Stanford Wine Ontology*⁴.

⁴<https://www.w3.org/TR/owl-guide/wine.rdf>

- **Cheminformatics domain:** evaluated against the *Chemical Information Ontology (CHEMINF)*⁵.
- **Environmental microbiology domain:** evaluated against the *AquaDiva ontology*⁶.
- **Sewer Networks domain:** evaluated against the *Sewer Network Ontology (SewerNet)*⁷.

5.3. LLMs

We apply the NeOn-GPT pipeline using four pretrained large language models. They were selected for their demonstrated effectiveness in knowledge extraction and ontology engineering tasks such as triple generation, schema induction, and concept hierarchy construction [59,97]. Their inclusion also supports a comparative analysis between proprietary and open-source models, which differ in their reasoning capacities and deployment constraints.

- **GPT (Proprietary):** an instruction-tuned model with strong reasoning and generation capabilities (GPT-4o).
- **Mistral (Open Source):** a compact, performant model offering a publicly accessible alternative (mistral-large).
- **Llama (Open Source):** a mixture-of-experts model with a 17B active parameter configuration, (llama-4-maverick-17B-128E-Instruct).
- **Deepseek (Open Source):** a model with sparse-attention mechanisms designed for long-context reasoning and improved efficiency (deepseek-v3.2-exp).

5.4. Experiment Setup

Each run begins with a domain description (120-150 words), a curated list of 10–15 domain-specific keywords, a few-shot examples, and a domain-specific persona (See Figure 1). The LLM is then guided through all NeOn-GPT stages. This process is repeated independently for each of the four selected domains, using the [four language models](#), resulting in sixteen ontologies (four domains × four LLMs).

6. Results and Discussion

6.1. Structural Analysis

The structural evaluation examines how effectively NeOn-GPT guides LLMs in constructing ontologies with coherent size, organization, and semantic structure, relative to established gold-standard ontologies. [Since LLMs were not provided with the same datasets or expert resources used to build the gold ontologies](#), the goal is not to reproduce them. Rather, this evaluation assesses whether the ontologies generated through our methodology exhibit appropriate scale and structural patterns characteristic of ma-

⁵<https://bioportal.bioontology.org/ontologies/CHEMINF>

⁶<https://www.aquadiva.uni-jena.de/>

⁷<http://sewernet.msem.univ-montp2.fr/>

Domain	Model	Axioms	Logical Axioms	Classes	Subclasses	Max. Depth
Wine	Gold	1046	889	138	228	3
	GPT-4o	4627	4296	55	704	1
	Mistral-large	890	582	144	124	3
	Llama-4-maverick-17B	750	376	51	15	2
	Deepseek-v3.2-exp	1982	1001	267	175	3
Cheminformatics	Gold	1651	494	349	370	5
	GPT-4o	1242	445	139	110	2
	Mistral-large	975	496	70	9	1
	Llama-4-maverick-17B	486	203	32	12	2
	Deepseek-v3.2-exp	1416	695	163	173	3
Environmental Microbiology	Gold	78840	16303	8892	14219	27
	GPT-4o	795	392	342	108	3
	Mistral-large	800	193	97	28	2
	Llama-4-maverick-17B	616	88	92	34	1
	Deepseek-v3.2-exp	2028	494	326	288	3
Sewer Networks	Gold	951	249	115	206	4
	GPT-4o	796	341	110	97	3
	Mistral-large	1187	615	141	168	3
	Llama-4-maverick-17B	871	385	114	69	2
	Deepseek-v3.2-exp	1686	642	130	156	4

Table 2. Comparison of structural size, coverage, and hierarchy of gold-standard and NeOn-GPT-generated ontologies across four domains. Subclasses correspond to the number of SubClassOf axioms extracted from Protégé.

ture domain ontologies. It is essential to note that this evaluation examines how LLMs structure the ontology without making claims about correctness or fidelity to a gold standard; that is, structural analysis does not assess whether the specific classes, properties, axioms, or constraints generated by the LLMs match those defined in the gold standard ontologies.

6.1.1. Structural Size

This evaluation examines **Step 3 Ontology Conceptualization (Section 3.1.3)**, which generates the core conceptual structure of an ontology, **its classes, axioms, and logical axioms**.

Table 2 and Appendix B.1: Figures 9–12 show a consistent pattern across all four domains: LLM-generated ontologies are structurally smaller than the gold standards, yet they preserve the relative scale of each domain. Large ontologies, such as Environmental Microbiology (8,892 classes and 78,840 axioms), yield the largest LLM outputs, while smaller domains yield proportionally smaller ones. This indicates that LLMs capture high-level signals of domain complexity even when they cannot reproduce its full structural depth.

In moderately sized domains such as Wine and Sewer Network, several models approximate the size of the gold **class** sets (e.g., Wine: 138 gold vs. 144 Mistral and 267

DeepSeek; Sewer Network: 115 gold vs. 141 Mistral and 130 DeepSeek). Logical axioms exhibit similar behavior (e.g., Sewer Network: 249 gold vs. 341 GPT-4o and 385 Llama-4). The Environmental Microbiology domain, with a gold-standard ontology comprising 8,892 classes and 16,303 logical axioms, reveals clearer limitations: the strongest model (DeepSeek) produces only 326 classes and 494 logical axioms. These differences indicate that LLMs effectively manage the conceptual footprint of small-to-mid-scale domains, but struggle to approximate the conceptual richness of large scientific ontologies.

Axiom and logical-axiom counts show a more nuanced pattern than class counts. In the smaller and medium-sized domains (Wine, Cheminformatics, Sewer Networks), LLM-generated ontologies are generally of the same order of magnitude as the gold standards and, in some cases, even exceed them. For example, in Wine, the gold ontology contains 1,046 axioms and 889 logical axioms, whereas GPT-4o produces 4,627 axioms and 4,296 logical axioms, and DeepSeek 1,982 and 1,001, respectively. In Cheminformatics, all models produce slightly fewer axioms than the gold standard (1,651), but Mistral and DeepSeek match or surpass the gold ontology in logical axioms (496 and 695 vs. 494). Sewer Networks shows a similar pattern, with several models generating more axioms and logical axioms than the gold ontology (e.g., DeepSeek: 1,686 axioms and 642 logical axioms vs. 951 and 249). This indicates that LLMs are capable of producing a substantial amount of formally structured OWL content, capturing relationships, constraints, and class axioms, even when their conceptual coverage does not fully match the gold standard. By contrast, Environmental Microbiology is a clear outlier: here, the gold ontology's 78,840 axioms and 16,303 logical axioms dwarf the LLM outputs (up to 2,028 axioms and 494 logical axioms), indicating that current models cannot approximate the axiomatic richness of very large scientific ontologies.

Model-specific tendencies further shape these outcomes. DeepSeek-v3.2 generally produces the richest structures among the models, but still falls far short of expert-curated axiom density in domains such as Environmental Microbiology. GPT-4o produces structurally smaller ontologies but does so in a balanced way, preserving the natural ratios between classes, axioms, and logical axioms found in the gold standard. Llama-4 often expands class sets (e.g., 385 classes in Sewer Network vs. 249 gold) without increasing logical axioms accordingly, producing broad but weakly constrained schemas. Mistral-large is the most volatile, under-generating in Cheminformatics (70 vs. 349 gold classes) while over-generating in Sewer Network (141 vs. 115).

6.1.2. Structural Hierarchy Characteristics

This evaluation examines Step 3 Ontology Conceptualization (Section 3.1.3), specifically the **hierarchy-building** activity, which organizes the identified classes into structured subclass relationships. In Wine, Cheminformatics, and Sewer Network Applications, LLMs approximate the gold taxonomy closely, typically within a single level, indicating that Step 3 supports the reconstruction of moderate domain hierarchies. Only Environmental Microbiology breaks this trend: its 27-level hierarchy collapses to 1–3 levels across models, indicating difficulty handling deeply nested biological taxonomies. DeepSeek achieves the highest depths across models, matching the gold depth in Wine and Sewer Network and surpassing all others in Cheminformatics and Environmental Microbiology.

Domain	Model	Obj. Prop.	Data Prop.	Inverse	Functional	Transitive	Symmetric	Reflexive
Wine	Gold	16	1	2	6	1	1	0
	GPT-4o	80	27	111	0	20	72	72
	Mistral-large	87	14	44	19	8	0	13
	Llama-4-maverick-17B	60	23	21	0	8	12	10
	Deepseek-v3.2-exp	159	7	55	5	5	5	5
Cheminformatics	Gold	53	6	11	0	4	5	8
	GPT-4o	82	20	32	12	8	8	0
	Mistral-large	93	66	36	0	0	0	0
	Llama-4-maverick-17B	25	17	10	5	5	5	5
	Deepseek-v3.2-exp	106	51	31	12	12	12	5
Environmental Microbiology	Gold	245	14	46	19	15	4	0
	GPT-4o	8	21	0	0	3	3	2
	Mistral-large	90	21	44	1	1	1	1
	Llama-4-maverick-17B	108	20	17	7	6	8	13
	Deepseek-v3.2-exp	150	59	13	17	3	11	10
Sewer Networks	Gold	22	0	0	0	0	1	0
	GPT-4o	53	54	20	6	5	5	6
	Mistral-large	46	42	10	0	10	0	0
	Llama-4-maverick-17B	85	34	16	0	2	10	11
	Deepseek-v3.2-exp	90	75	15	4	8	5	11

Table 3. Comparison of structural property characteristics and detailed object property characteristics (inverse, functional, transitive, symmetric, reflexive), metrics are extracted using Protégé [70].

6.1.3. Structural Property Characteristics

Step 6 Ontology Formal Modeling (Section 3.1.6) is reflected in the generation of object and data properties. Object property counts and data property counts are presented in [Table 2](#) and [Appendix B.1](#). LLMs often expand the **object-property** space, most clearly in Wine (16 gold vs. 80–159 LLM) and Sewer Network (22 gold vs. 46–90 LLM), but under-generate in Environmental Microbiology (245 gold vs. 8–150 LLM), where biological relation structures are more intricate. **Data properties** show a uniform trend: all models generate more than the gold ontology across all domains (e.g., Wine: 1 gold vs. 7–27 LLM; Environmental Microbiology: 14 gold vs. 21–59 LLM).

Table 3 presents a detailed **object-property** analysis, showing how well models assign semantic behaviors (**inverse, functional, transitive, symmetric, reflexive**) to the object properties they generate.

Inverse properties are the most difficult characteristic across all domains. In domains with modest gold inverse counts, Wine (2) and CHEMINF (11), LLMs routinely overshoot (e.g., GPT-4o with 111 in Wine and 32 in CHEMINF). Conversely, in Environmental Microbiology, where the gold ontology has 46 inverse relations, only Mistral (44) approaches the gold scale; all others massively under-generate (0–17). This inconsistent behavior, characterized by both over- and under-projection, suggests that inverse detection is the least reliably learned characteristic.

Functional properties are the most systematically under-produced. Gold standards contain very few (e.g., 6 in Wine, 0 in Cheminformatics and Sewer Network, 19 in Environmental Microbiology), and LLMs mirror this pattern: even when they generate large numbers of object properties overall, functional properties remain nearly absent relative to total property volume (e.g., GPT-4o produces only 12 in Cheminformatics). This is not necessarily a limitation of LLMs, but rather a modeling pattern: functional properties encode strict cardinality constraints that are used sparingly in expert ontologies due to their rigidity and the strong assumptions they impose.

Transitive and symmetric properties show moderate but domain-dependent recovery. LLMs exceed gold transitivity in several domains (e.g., DeepSeek’s 12 transitive properties in Cheminformatics vs. 4 gold), yet drastically underperform in Environmental Microbiology (gold = 15; models = 1–6). Similarly, symmetric properties are over-generated in small domains (e.g., GPT-4o’s 72 vs. 1 gold in Wine) but inconsistently reproduced in complex ones. These trends suggest that LLMs capture transitivity and symmetry more effectively in domains where these patterns are broadly applicable; however, their performance declines when such properties are tied to more domain-specific modeling conventions rather than general relational regularities.

Reflexive properties are rare even in the gold standards. Reflexivity appears only in the Cheminformatics ontology (8 reflexive properties), and none of the other domains include any. LLMs nevertheless generate small numbers of reflexive relations across several domains (e.g., 2–13 in Environmental Microbiology; 6–11 in Sewer Networks).

6.1.4. Schema-level Domain and Range Assertions

Schema-level constraints generated in **Step 6 Ontology Formal Modeling (Section 3.1.6)** are reflected in the **domain and range assertions** reported in Table 4. Across all domains, LLM outputs scale proportionally with the number of objects and data properties they generate (Table 2), indicating that models reliably follow the instructions and few-shot examples provided in the NeOn-GPT prompts. For example, in the Wine ontology, models that create between 60 and 159 object properties and 14 and 27 data properties also produce 82 and 167 corresponding domain and range assertions. A similar pattern appears in Cheminformatics, where producing 82–106 object properties and 20–66 data properties leads to 103–137 domain and range assertions.

These patterns show that LLMs are capable of specifying the associated schema constraints when generating properties. Although this does not guarantee semantic correctness, the tight coupling between property generation (Table 2) and schema assertion generation (Table 4) demonstrates that LLMs systematically apply domain/range typing rules from the prompts, rather than hallucinating unconstrained properties. Across domains, DeepSeek, typically the model with the largest property counts, also produces the most domain and range assertions, while GPT-4o and Mistral show more moderate scaling. This alignment suggests that NeOn-GPT effectively enforces the relational scaffolding expected in Step 6, and that constraint generation behavior is primarily driven by each model’s propensity to generate properties in the first place.

6.1.5. Instance Population Characteristics

Step 7 Ontology Population (Section 3.1.7) is reflected in the number of **individuals** (Table 4). In domains with substantial instance sets such as Wine (206) and Environmen-

Domain	Model	Domain Assertions	Range Assertions	Individuals	Annotations
Wine	Gold	11	14	206	3
	GPT-4o	104	104	27	142
	Mistral-large	102	102	117	242
	Llama-4-maverick-17B	82	86	72	276
	Deepseek-v3.2-exp	132	167	177	659
Cheminformatics	Gold	29	27	0	823
	GPT-4o	103	103	35	559
	Mistral-large	121	119	79	353
	Llama-4-maverick-17B	27	30	46	213
	Deepseek-v3.2-exp	134	137	73	464
Environmental Microbiology	Gold	161	165	95	53101
	GPT-4o	21	21	22	269
	Mistral-large	70	70	90	248
	Llama-4-maverick-17B	64	67	65	215
	Deepseek-v3.2-exp	199	214	100	627
Sewer Networks	Gold	7	10	7	548
	GPT-4o	85	85	33	252
	Mistral-large	79	79	117	344
	Llama-4-maverick-17B	85	87	83	355
	Deepseek-v3.2-exp	157	161	30	765

Table 4. Comparison of schema-level domain, range, individuals, and annotation assertions (including labels, comments, and other metadata annotations). Metrics extracted using Protégé [70].

tal Microbiology (95), some models generate comparable populations (e.g., DeepSeek: 177 and 100), while others produce far fewer (e.g., GPT-4o: 27 and 22). In schema-only domains, such as Cheminformatics, all models introduce new individuals (35–79), indicating a tendency to generate illustrative examples. Higher LLM counts are not necessarily problematic—many gold ontologies rely on imported vocabularies for instances, so richer populations may reflect examples that the imports otherwise provide.

6.1.6. Annotation and Documentation Characteristics

Step 8 Ontology Documentation (Section 3.1.8) is reflected in the total number of **annotation assertions (labels, comments, and metadata)** in Table 4. Across all domains, LLMs produce documentation in proportion to the overall size of the ontologies they generate (Table 2). Smaller ontologies receive modest documentation assertions, while larger ones are accompanied by a substantially larger number of documentation assertions. In Wine, for example, LLM-generated ontologies remain structurally small (55–267 classes, 27–177 individuals), and models produce 142–659 annotations. In domains where models generate comparatively larger schemas, such as DeepSeek in Environmental Microbiology (326 classes, 150 object properties), documentation also increases markedly (627 annotations). These patterns suggest that LLMs adhere to the

NeOn-GPT prompt structure and few-shot examples, with a focus on human-readable labels and descriptions.

6.1.7. *Structural Composition and Logic Distribution*

Beyond overall size, the internal composition of axioms reveals systematic differences in how LLMs formalize knowledge. While gold-standard ontologies typically allocate a substantial portion of their logical axioms to class hierarchies (TBox), LLM-generated ontologies tend to allocate a comparatively higher proportion of axioms to object-property definitions and property characteristics (RBox), alongside a lower number of subclass axioms. This pattern is visible across domains in Tables 2–4, where relatively shallow hierarchies (Section 3.1.3) coexist with large numbers of object properties and schema-level constraints (Section 3.1.6).

This distribution suggests that LLMs preferentially encode domain structure through relations between classes rather than through deep taxonomic organization. In practical terms, knowledge is represented using a larger number of predicates and constraints over a flatter class structure, rather than through multi-level subsumption hierarchies. The effect is most pronounced in large domains, such as Environmental Microbiology, where the gold ontology’s deep taxonomy is not recovered; however, object-property counts and associated domain and range assertions still scale with model output.

In addition, LLM-generated ontologies consistently contain a higher proportion of annotation axioms relative to logical axioms (Table 4), indicating a tendency to emphasize natural-language descriptions alongside formal structure. Taken together, these results show that LLMs can produce ontologies that are rich in relational structure and documentation, but comparatively limited in hierarchical depth, reflecting differences in how domain knowledge is organized rather than an attempt to replicate gold-standard designs.

A closer inspection of the internal distribution of schema axioms further clarifies how this structural flattening manifests. While LLM-generated ontologies often contain substantial numbers of logical axioms, these axioms are not distributed in a manner that reproduces the hierarchical organization of expert-built ontologies. In particular, Table 2 shows that, even when models generate hundreds or thousands of logical axioms, this does not translate into increased hierarchical depth. This disconnect is most evident in Environmental Microbiology, where the gold-standard ontology reaches a maximum depth of 27, whereas all LLM-generated ontologies remain confined to depths between 1 and 3 despite producing up to 494 logical axioms (DeepSeek).

This pattern indicates that LLMs tend to allocate logical expressivity toward shallow subclass structures and non-hierarchical schema axioms rather than toward deep terminological refinement. In other words, logical axioms are predominantly used to describe properties, constraints, and relationships at a single or limited number of abstraction levels, rather than to incrementally specialize concepts across multiple layers of a taxonomy. As a result, increases in logical axiom counts do not yield proportional increases in taxonomic depth.

The effect is less pronounced in smaller domains, such as Wine and Sewer Networks, where the gold-standard hierarchies themselves are shallow (with depths of 3 and 4, respectively). In these cases, several models match the gold depth exactly, suggesting that NeOn-GPT supports the recovery of modest terminological hierarchies when domain

taxonomies are limited in depth. However, when expert ontologies rely on deeply nested conceptual structures, LLM-generated ontologies instead favor breadth-oriented schema construction over hierarchical refinement.

6.1.8. *Model-Specific Structural Tendencies*

The comparative performance of the four large language models across domains reveals distinct structural tendencies that align with different stages of the ontology engineering process. Rather than identifying a single best-performing model, the results highlight complementary strengths and trade-offs in how models allocate expressivity across terminological, relational, and assertional components when guided by the NeOn-GPT pipeline.

GPT-4o exhibits strong performance in schema-level relational modeling and broad conceptual coverage. Across multiple domains, including Wine and Sewer Networks, GPT-4o generates large numbers of object and data properties accompanied by proportionally high numbers of domain and range assertions (Table 4), indicating consistent grounding of properties within class hierarchies. At the same time, GPT-4o produces comparatively shallow class hierarchies (Table 2), suggesting that its expressivity is concentrated in relational structure rather than in deep terminological refinement. This profile makes GPT-4o particularly suitable for early-stage domain exploration and broad schema construction, where coverage and relational scaffolding are prioritized over taxonomic depth.

DeepSeek-v3.2-exp consistently generates the structurally richest ontologies among the evaluated models. It produces the largest numbers of classes, properties, domain and range assertions, individuals, and annotation axioms across domains (Tables 2-4). DeepSeek also achieves the greatest hierarchy depth among LLM-generated ontologies, especially in technically complex domains such as Environmental Microbiology, although these depths remain substantially lower than those observed in gold-standard ontologies. Its tendency to generate comparatively large instance populations suggests that DeepSeek is well suited for producing candidate knowledge graph content or illustrative instance data, provided that appropriate validation and refinement mechanisms are applied.

Llama-4-maverick-17B and Mistral-large exhibit more localized and domain-dependent behavior. Both models tend to generate more compact schemas than DeepSeek, with fewer classes, properties, and individuals, while still maintaining coherent structural organization. Llama-4-maverick-17B often produces restrained instance populations and shallow hierarchies, whereas Mistral-large shows greater variability across domains, under-generating in some cases and over-generating in others. These characteristics make both models suitable for lightweight schema construction and terminology prototyping in domains where interpretability and structural simplicity are preferred.

Taken together, the results indicate that different models emphasize different aspects of ontology structure, including relational schema construction, instance population, and lightweight conceptual modeling, rather than exhibiting uniform behavior across all modeling tasks.

6.2. *Lexical Analysis*

The lexical evaluation assesses the extent to which NeOn-GPT directs LLMs in generating terminology that aligns with the conceptual vocabulary used in gold-standard on-

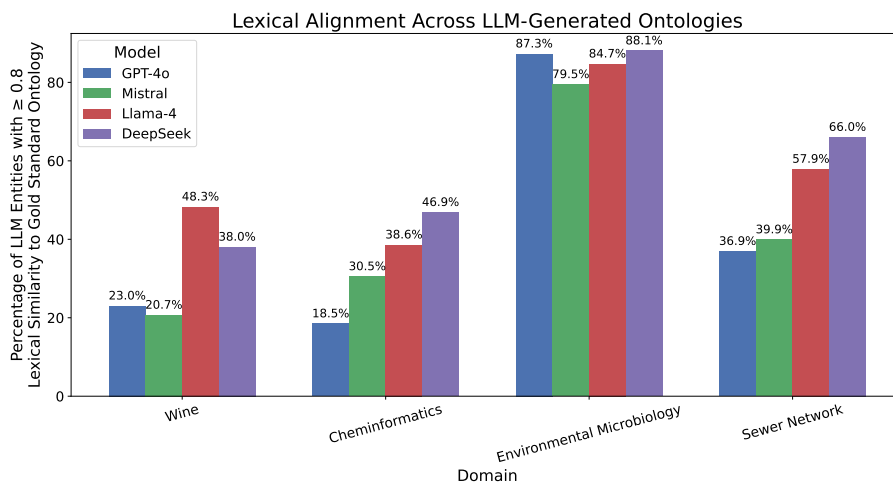


Figure 4. Percentage of entities and properties in LLM-generated ontologies whose entities and properties achieve a similarity score ≥ 0.8 when compared to the corresponding gold-standard ontology. Results are shown for four domains (Wine, Cheminformatics, Environmental Microbiology, and Sewer Network) and four LLMs (GPT-4o, Mistral, Llama-4, and DeepSeek).

ologies. We measure (i) exact lexical matches Appendix B.2: Table 6) and (ii) high-similarity matches using Jaro–Winkler similarity ≥ 0.8 (Figure 6.2). Detailed cross-ontology similarity matrices are provided in Appendix B.2: Figures 17-32.

Across domains, Environmental Microbiology exhibits the strongest lexical alignment, with similarity scores ranging from 79.5 % (Mistral) to 88.7 % (DeepSeek). This trend reflects the accessibility of the domain’s terminology, which includes widely used scientific terms such as (e.g., "GroundWater", "Temperature", "Species", "Habitat"), and examples of these terms appear in Table 6. This domain also yields one of the largest sets of exact matches across all models, particularly for DeepSeek and Mistral. Beyond these exact correspondences, many additional labels achieve high string-based similarity scores (≥ 0.8), contributing to the overall alignment observed in Figure 6.2.

In contrast, Cheminformatics consistently yields the weakest lexical overlap, with similarity scores ranging from 18.5 % (GPT-4o) to 46.9 % (DeepSeek) and no exact matches across any model. The absence of exact matches reflects the highly specialized nature of CHEMINF terminology, which often uses formal identifiers such as "CHEMINF_000373", technical descriptors, or compound naming conventions rarely encountered in natural language sources. As a result, the LLMs generate plausible chemical or methodological terms, but these do not lexically converge on the curated CHEMINF vocabulary. This domain, therefore, highlights a fundamental challenge for LLMs: domains in which the gold-standard ontology uses formalized or ontology-specific naming conventions are significantly harder to approximate lexically.

The Wine domain shows intermediate levels of lexical alignment. In the Wine domain, Llama-4 and DeepSeek achieve the highest similarity (48.3 % and 38.0 %), supported by a moderate number of exact matches (e.g., "RedWine", "WhiteWine", "Wine", "Vintage", "hasFlavor") (See Table 6). GPT-4o and Mistral achieve

lower similarity scores, consistent with their smaller sets of exact matches. The results suggest that wine vocabulary, though domain-specific, is widely represented in general discourse, enabling partial alignment even when the specific gold terminology is not fully reproduced.

The Sewer Network domain shows similar intermediate behavior. Despite limited exact matches in Table 6, lexical similarity scores range from 36.9 % (GPT-4o) to 66.0 % (DeepSeek). This is supported by the presence of frequently used infrastructure terms, such as "manhole", "pipe", "inspection", and "repair", examples of which appear in Table 6. These terms occur commonly in technical and engineering texts, which makes them more likely to be reproduced by LLMs.

Finally, model-level trends appear consistently across domains. DeepSeek achieves the highest lexical similarity in three out of four domains, reflecting its tendency to generate a wider and more domain-aligned vocabulary. Llama-4 also performs strongly, especially in the Wine and Sewer Network domains. GPT-4o produces more conservative vocabularies, resulting in fewer exact matches and lower similarity in domains where diversity of naming is required, though its outputs remain coherent and domain-relevant. Mistral exhibits the most variable performance across domains, achieving a reasonable level of similarity in Environmental Microbiology but substantially lower alignment in Wine and Cheminformatics.

Taken together, these findings demonstrate that lexical alignment in NeOn-GPT-generated ontologies is strongly dependent on both the conceptual system of the domain and the LLM pre-training, which influence its output and lexical diversity. Domains with broadly used scientific or infrastructural vocabularies enable strong alignment, whereas formalized or ontology-specific naming schemes remain a significant challenge for current LLMs. Occasional patterns of partial similarity observed in the appendix heatmaps support these conclusions (See Figures 17-32).

6.3. Semantic Analysis

Semantic evaluation measures the extent to which NeOn-GPT-generated ontologies capture conceptual rather than lexical correspondence with gold-standard ontologies. Using the entity- and triple-level embedding framework described in Section 4, we assess whether LLMs reproduce domain semantics through structurally meaningful and contextually aligned concepts and relations.

Overall, the results show that LLMs achieve meaningful semantic alignment with gold-standard ontologies, with the majority of entities and schema-level triples clustering in the 0.5–0.8 cosine-similarity range (See Figures 5-8). This indicates that LLMs can approximate the conceptual neighborhoods of many domain entities, even when lexical forms differ. Although very high-similarity matches ($\geq 80\%$) are less frequent, their presence across all domains demonstrates that LLMs are capable of capturing core domain semantics, while the broader 0.5–0.8 % distribution reflects stable partial alignment rather than random or inconsistent behaviour.

Clear domain-level trends emerge. Environmental Microbiology exhibits the strongest semantic alignment, with DeepSeek and Llama achieving the largest 60–80 % and 80–90 % buckets (See Figure 7). This reflects the fact that biological terminology and environmental processes are more widely represented in the training corpora of LLMs, enabling partial reconstruction of domain semantics even without exposure to specialized datasets.

Wine and Sewer Networks show moderate alignment, consistent with domains grounded in widely documented real-world concepts (See Figures 5, 8). In contrast, Cheminformatics yields the weakest semantic similarity: the gold standard ontology relies heavily on formal identifiers and chemistry-specific constructs (e.g., CHEMINF codes) unavailable to LLMs, leading to widespread $\leq 60\%$ matches despite otherwise coherent generated terminology (See Figure 6).

Model-level differences are also consistent. DeepSeek achieves the strongest alignment across all domains, producing the highest proportion of 70–80% and 80–90% matches for both entities and triples. Llama-4 performs comparably in linguistically accessible domains but lags in more technical ones. GPT-4o shows stable but conservative alignment, with fewer high-similarity peaks. Mistral is the most variable, performing well in biology but poorly in Wine and Cheminformatics.

These findings reveal a distinction between entity-level and triple-level semantic behaviour in LLM-generated ontologies. Entity embeddings consistently achieve higher similarity than triple embeddings, indicating that LLMs capture the conceptual meaning of individual classes and properties more reliably than the relational structures linking them. Triple-level similarity, while following the same overall patterns, exhibits a spread toward lower similarity buckets—reflecting the greater difficulty of reconstructing schema-level relations such as subclass paths, domain–range patterns, and other formal constraints. Nevertheless, the presence of numerous mid- to high-similarity triples across all domains demonstrates that LLMs are capable of recovering substantial portions of the underlying semantic structure when provided with concise domain descriptions.

7. Ablation Study

We conducted an ablation study of the NeOn-GPT pipeline to assess the contribution of the verification and resolution stage to the generation of viable ontologies. This stage comprises (i) Turtle syntax validation using RDFLib, (ii) logical consistency checking using OWL reasoners, and (iii) modeling-pitfall detection using OOPS!. The goal of this ablation is to quantify the effect of removing this stage on syntactic validity, logical consistency, and the presence of critical modeling pitfalls, while keeping all other components of the pipeline unchanged.

7.1. Ablation Setting

We compare two configurations:

- Full NeOn-GPT: the complete NeOn-GPT pipeline, including ontology draft generation followed by validation and resolution using RDFLib, OWL reasoners, and OOPS!.
- NeOn-GPT (–Verification/Resolution): an ablated variant in which the verification and resolution stage is entirely removed. In this configuration, the ontology produced at the end of the ontology draft generation stage is considered final, without undergoing detection or resolution of syntax errors, logical inconsistencies, or modeling pitfalls.

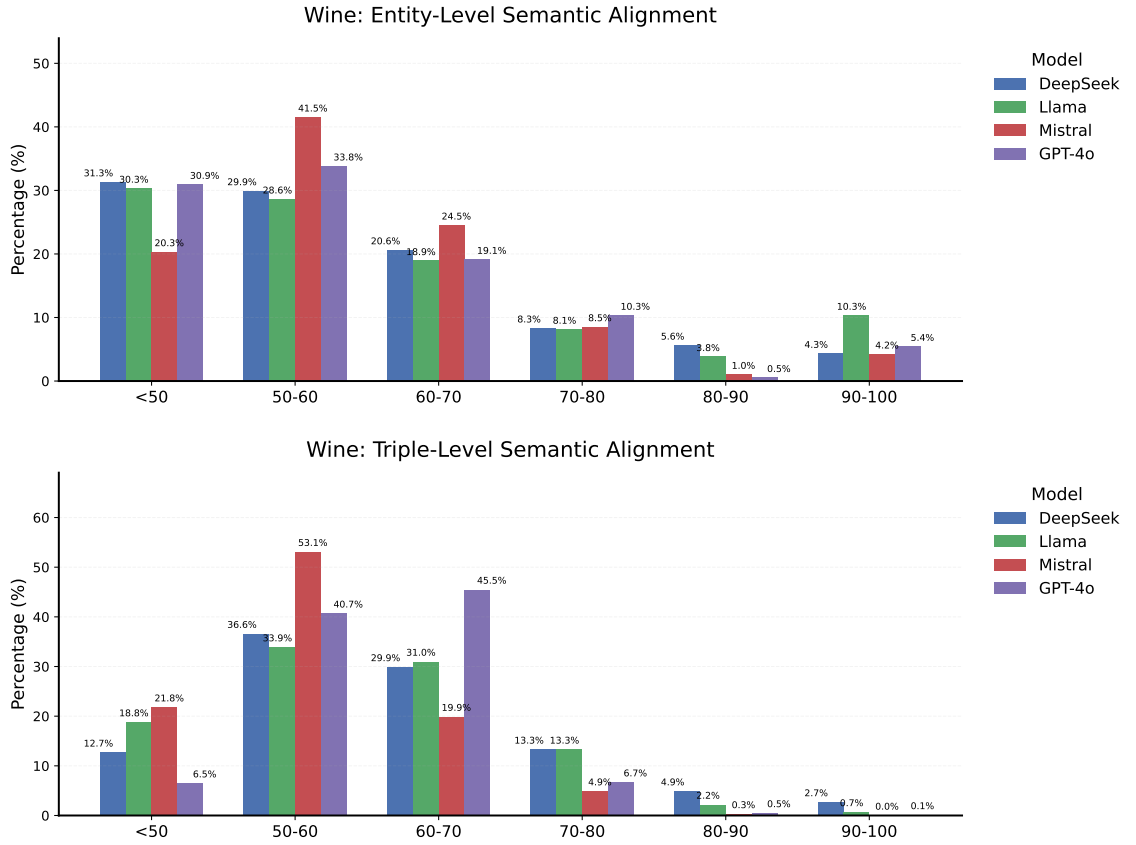


Figure 5. Entity-level (top) and triple-level (bottom) semantic alignment of LLM-generated ontologies in the Wine domain.

All other aspects of the pipeline are held constant across configurations, including prompts, domain descriptions, few-shot examples, ontology reuse strategies, and large language model settings.

The ablation is evaluated on the full set of sixteen generated ontologies (four domains \times four LLMs). For each ontology and each configuration, we measure the following indicators:

- **Syntax errors:** whether the ontology contains Turtle syntax errors when parsed.
- **Logical inconsistencies:** whether the ontology is reported as inconsistent by OWL reasoners.
- **Critical modeling pitfalls:** the number and type of critical pitfalls detected by OOPS!.

Each indicator is measured independently for the Full and –Validation/Resolution configurations. No additional filtering, repair, or post-processing is applied in the ablated setting.

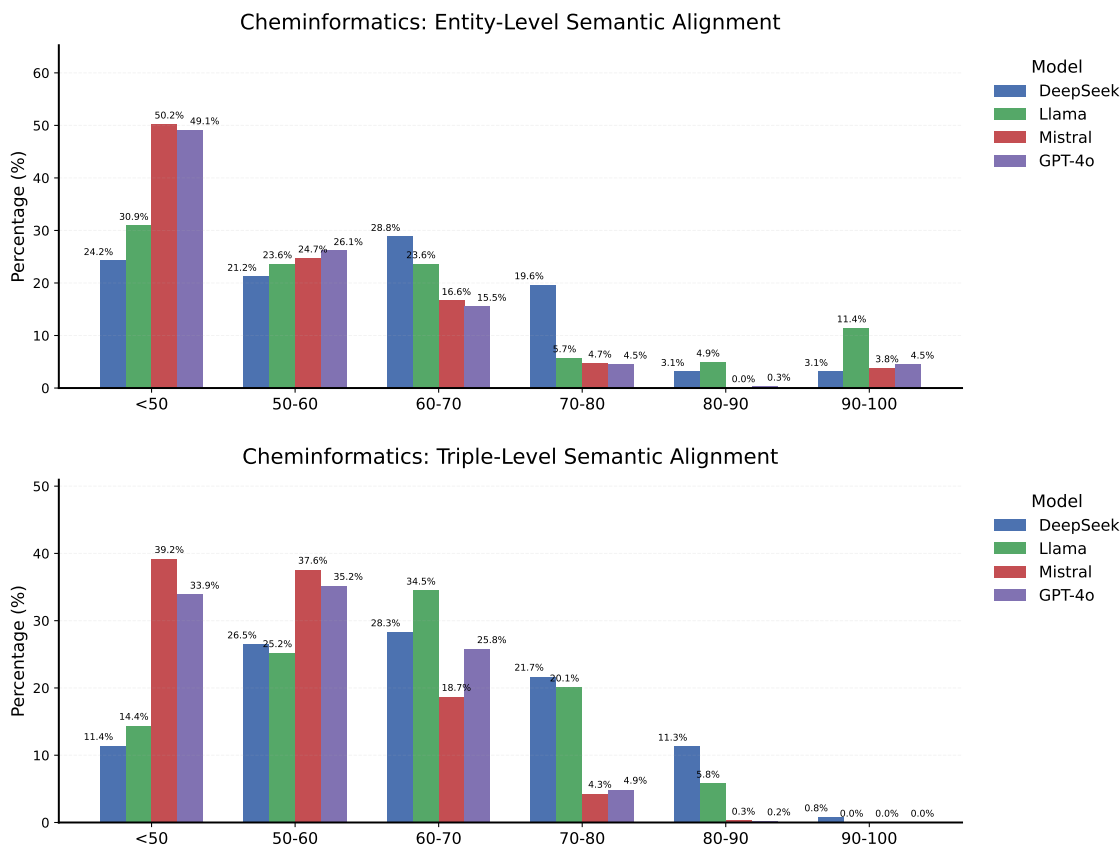


Figure 6. Entity-level (top) and triple-level (bottom) semantic alignment of LLM-generated ontologies in the Cheminformatics domain.

7.2. Ablation Results

Table 5 summarizes the impact of removing the verification and resolution stage across all sixteen ontologies. The results reveal systematic differences between ontology drafts produced without this stage and the final validated ontologies, across all three evaluated dimensions.

In terms of syntactic validity, ontology drafts generated without verification and resolution exhibit non-trivial numbers of Turtle syntax errors across most domain-model combinations. The number of syntax errors ranges from 2 to 14 in three of the four domains, with the highest counts observed for Llama-4-maverick-17B in Environmental Microbiology (14 errors) and Cheminformatics (12 errors), and for Mistral-large and GPT-4o in Sewer Networks (up to 7 and 5 errors, respectively). After applying the verification and resolution stage, syntax errors are fully eliminated in all cases, yielding syntactically valid ontologies for all domains and models.

Logical consistency shows a similarly clear separation between the two configurations. Without verification and resolution, logical inconsistencies occur in 7 out of the 16 generated ontologies, spanning all domains except Environmental Microbiology. These

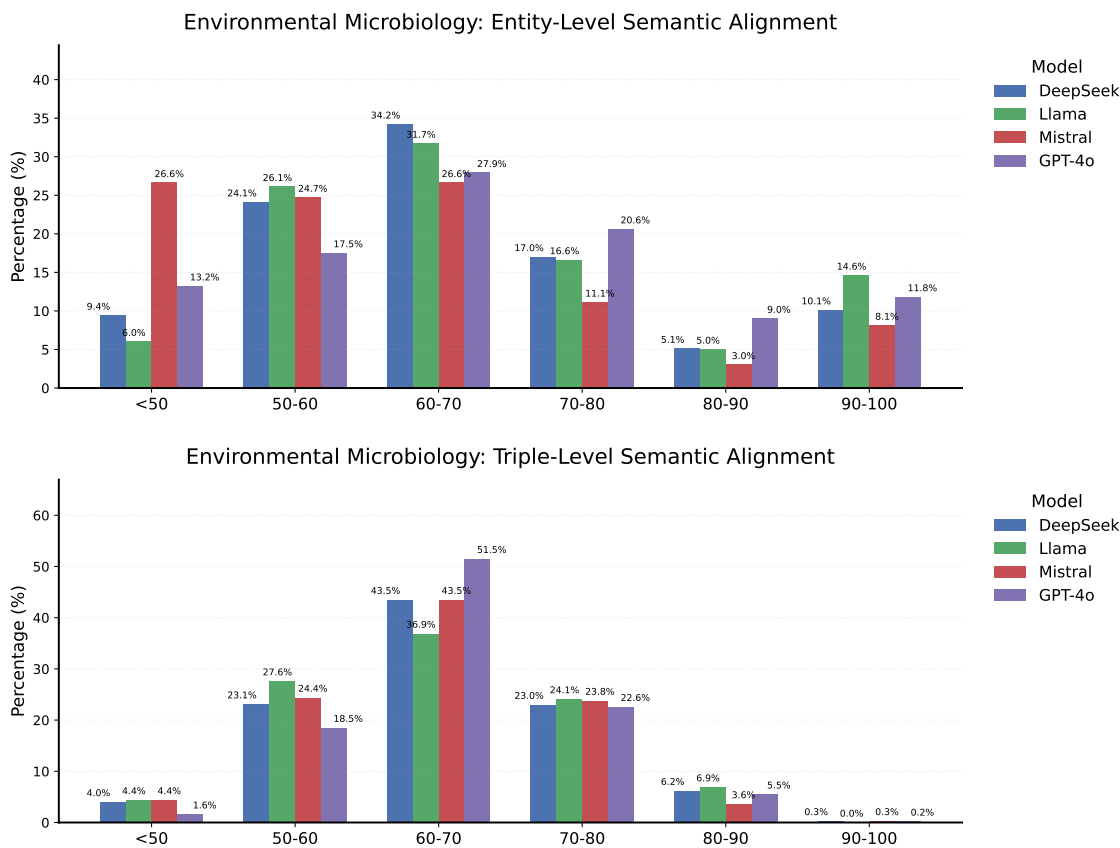


Figure 7. Entity-level (top) and triple-level (bottom) semantic alignment of LLM-generated ontologies in the Environmental Microbiology domain.

inconsistencies are not confined to a single model or domain, appearing, for example, in Cheminformatics and Sewer Networks across multiple models, and in the case of Wine, for Llama-4-maverick-17B. Following validation and resolution, all ontologies are reported as logically consistent, indicating that the resolution stage systematically addresses reasoning-level violations introduced during the draft generation stage.

The most pronounced differences appear in the analysis of critical modeling pitfalls. In the ablated configuration, critical pitfalls are detected in 13 out of the 16 ontologies, with counts ranging from 1 to 5 per ontology. These pitfalls span multiple OOPS! categories, most frequently including P05 (inverse relationships not explicitly declared), P19 (missing domain or range), and P28/P29 (incorrect or missing property constraints), with additional domain- and model-specific occurrences such as P06, P27, P31, and P39. Notably, even when no logical inconsistency is reported, critical modeling pitfalls are still present, indicating that syntactic and logical validity alone are insufficient to guarantee modeling quality. Formal definitions of all critical OOPS! pitfalls referenced in this analysis are provided in Appendix B.2.

After applying the verification and resolution stage, no critical modeling pitfalls are detected in any of the final ontologies. This holds across all domains and models, in-

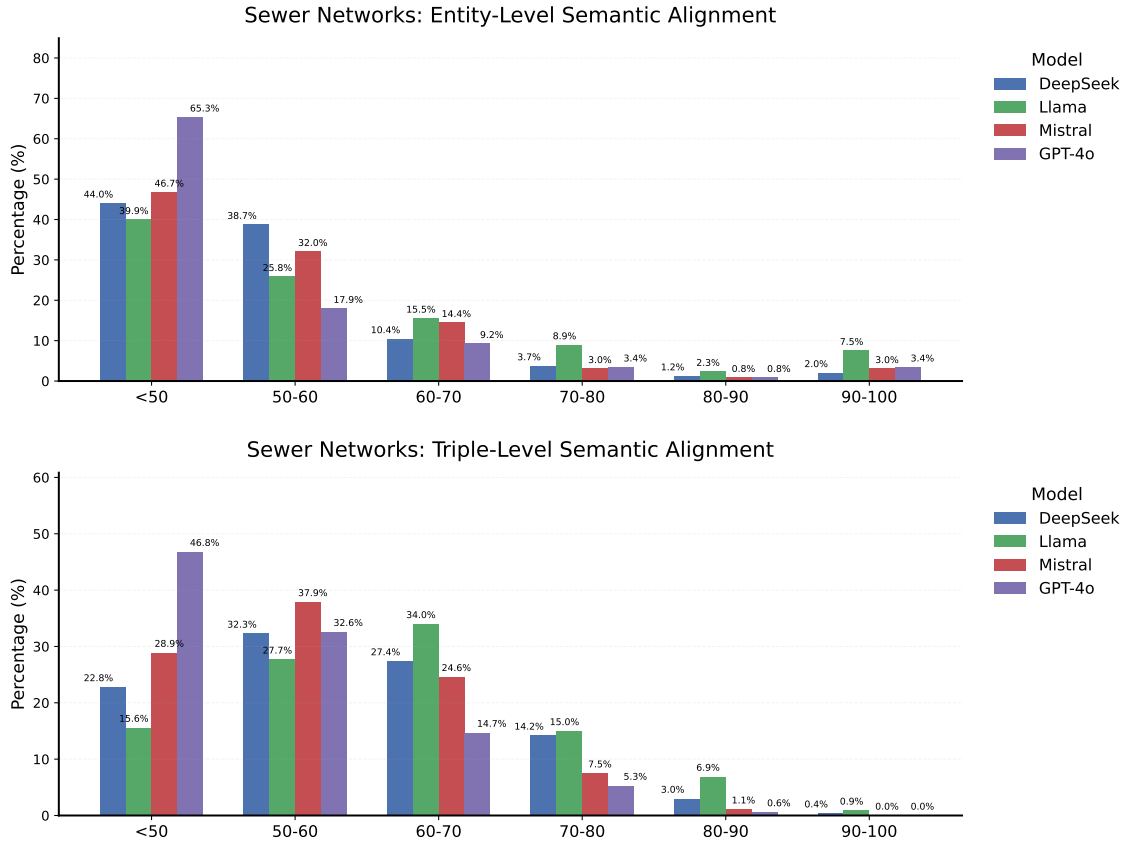


Figure 8. Entity-level (top) and triple-level (bottom) semantic alignment of LLM-generated ontologies in the Sewer Networks domain.

cluding cases where multiple distinct pitfall types co-occur in the ablated setting. Taken together, these results indicate that the verification and resolution stage plays a decisive role not only in enforcing syntactic correctness and logical consistency but also in systematically eliminating high-severity modeling defects that persist across domains and language models.

8. Limitations

A first limitation of the NeOn-GPT pipeline concerns the nature and scope of the inputs provided to the LLMs. For each domain, the models receive only a concise natural-language description (120–150 words), a short list of 10–15 domain-specific vocabulary terms, at most six few-shot examples, and along with the extracted fragments of existing ontologies relevant to the domain for reuse (See Section 3.1 and Figure 2). This input is substantially more limited than the resources typically available to ontology engineers when constructing gold-standard ontologies, which often include extensive domain corpora, controlled vocabularies, user stories, detailed requirement specifications,

Table 5. Ablation study results for NeOn-GPT on 16 ontologies (4 domains \times 4 LLMs). Results are reported before (w/o) and after (w/) the validation and resolution stage. For critical modeling pitfalls, we report both the total number of detected critical pitfalls (#) and the corresponding OOPS! pitfall codes (types). **C** and **I** indicate consistent and inconsistent ontologies, respectively.

Domain	Model	Syntax errors		Consistency		Critical pitfalls			
		w/o	w/	w/o	w/	w/o #	w/o types	w/ #	w/ types
Wine	GPT-4o	4	0	C	C	5	P05, P27, P28, P29, P31	0	–
	Mistral-large	3	0	C	C	0	–	0	–
	Llama-4-maverick-17B	11	0	I	C	5	P05, P19, P28, P29, P31	0	–
	Deepseek-v3.2-exp	2	0	C	C	4	P05, P06, P19, P29	0	–
Cheminformatics	GPT-4o	4	0	I	C	3	P05, P19, P29	0	–
	Mistral-large	6	0	I	C	3	P05, P19, P39	0	–
	Llama-4-maverick-17B	12	0	C	C	2	P19, P28	0	–
	Deepseek-v3.2-exp	0	0	I	C	3	P05, P19, P28	0	–
Environmental Microbiology	GPT-4o	2	0	C	C	0	–	0	–
	Mistral-large	4	0	C	C	1	P19	0	–
	Llama-4-maverick-17B	14	0	C	C	2	P05, P19	0	–
	Deepseek-v3.2-exp	0	0	C	C	3	P05, P19, P28	0	–
Sewer Networks	GPT-4o	5	0	C	C	2	P05, P29	0	–
	Mistral-large	7	0	I	C	2	P05, P28	0	–
	Llama-4-maverick-17B	3	0	I	C	1	P19	0	–
	Deepseek-v3.2-exp	4	0	I	C	3	P05, P19, P29	0	–

and multi-source documentation. Consequently, the LLM-generated ontologies tend to underperform on structural metrics, producing smaller schemas, fewer axioms, and hierarchies with limited depth, because the models are not exposed to the full depth of domain knowledge used to create the reference ontologies.

A second limitation is the reliance on manual effort for preparing the domain descriptions, few-shot examples (including fragments extracted from domain-relevant ontologies for reuse), and domain-specific keywords. While these inputs are comparatively lightweight, they may increase the configuration cost, which can affect scalability.

A third limitation concerns reproducibility under stochastic generation. In our experiments, we report one ontology generation per model and domain using fixed prompts and deterministic verification tools. While this supports the reproducibility of the pipeline execution, it does not quantify run-to-run variance in the generated ontologies.

Finally, our verification-and-resolution loop (syntax parsing, reasoner-based consistency checking, and OOPS! pitfall detection) ensures that evaluated ontologies are syntactically valid, logically consistent, and free of *Critical* and *Important* pitfalls. However, these checks do not guarantee conceptual completeness or alignment with all expert modeling choices, particularly in large domains with deep taxonomic structures. We also acknowledge the potential for bias or ontology drift introduced by the LLMs’ pretraining data and by the selection of reused material; our grounding and validation steps can mitigate some issues but cannot eliminate them entirely.

9. Conclusion and Future Work

The combined structural, lexical, and semantic analysis demonstrates that LLMs can generate ontologically relevant concepts and properties. However, they do not reliably replicate the size, hierarchical depth, or specific naming conventions of established expert-curated ontologies. They tend to produce smaller structures but often generate richer

properties. Based on the findings and limitations identified in this study, several promising directions for future research emerge that could advance the field of LLM-based ontology generation.

First, the current pipeline relies on expert-crafted examples for few-shot prompting and domain-specific natural language descriptions, which can increase the manual effort required to adapt the pipeline to new domains. With recent advances in LLM retrieval and agent-based systems, future work could integrate retrieval augmented generation (RAG) to automatically source relevant few-shot examples (and candidate reuse fragments) from curated knowledge bases, domain corpora, or ontology repositories. Alternatively, a multi-agent architecture could delegate few-shot example generation and LLM-generated persona validation to specialized agents, reducing the dependency on human experts while maintaining domain alignment.

Second, future work should evaluate robustness under stochastic generation by performing multi-run experiments per model and domain. This requires reliable ontology-level alignment, consolidation, and conflict resolution across independently generated runs.

Third, quantitative assessment of practitioner-level impact remains an open step. While prior user studies on LLM-assisted ontology engineering (e.g., OntoChat [47]) suggest that engineers may prefer delegating early-stage tasks such as requirements elicitation and competency question generation to LLM-based systems, we do not measure time or effort reduction in this work. A controlled user study across both simple and complex domains is an important next step in evaluating whether NeOn-GPT accelerates or simplifies ontology construction in practice.

Finally, extending evaluation beyond single gold standards is a promising direction. Comparing generated ontologies against multiple expert-curated ontologies per domain would enable analysis of inter-expert variation and perspective-dependent modeling choices, and would clarify how LLM-generated artifacts align with alternative expert conceptualizations. We also emphasize that deployment in sensitive domains (e.g., biomedical or environmental knowledge) should remain expert-supervised, and future work should further investigate risks of bias, drift, and attribution in AI-assisted ontology generation.

References

- [1] A. Hogan, E. Blomqvist, M. Cochez, C. d'Amato, G.d. Melo, C. Gutierrez, S. Kirrane, J.E.L. Gayo, R. Navigli, S. Neumaier et al., Knowledge graphs, *Synthesis Lectures on Data, Semantics, and Knowledge* **12**(2) (2021), 1–257.
- [2] N. Guarino, D. Oberle and S. Staab, What Is an Ontology?, in: *Handbook on Ontologies*, S. Staab and R. Studer, eds, International Handbooks on Information Systems, Springer, 2009, pp. 1–17. doi:10.1007/978-3-540-92673-3_0.
- [3] L.M. Schriml, C. Arze, S. Nadendla, Y.W. Chang, M. Mazaitis, V. Felix, G. Feng and W.A. Kibbe, Disease Ontology: a backbone for disease semantic integration, *Nucleic Acids Res.* **40**(Database–Issue) (2012), 940–946. doi:10.1093/NAR/GKR972. <https://doi.org/10.1093/nar/gkr972>.
- [4] P.L. Buttigieg, N. Morrison, B. Smith, C.J. Mungall and S.E. Lewis, The environment ontology: contextualising biological and biomedical entities, *J. Biomed. Semant.* **4** (2013), 43. doi:10.1186/2041-1480-4-43.
- [5] B. Smith, M. Ashburner, C. Rosse, J. Bard, W. Bug, W. Ceusters, L.J. Goldberg, K. Eilbeck, A. Ireland, C.J. Mungall et al., The OBO Foundry: coordinated evolution of ontologies to support biomedical data integration, *Nature biotechnology* **25**(11) (2007), 1251–1255.

- [6] P. Casanovas, N. Casellas and J. Vallbé, An Ontology-Based Decision Support System for Judges, in: *Law, Ontologies and the Semantic Web - Channelling the Legal Information Flood*, J. Breuker, P. Casanovas, M.C.A. Klein and E. Francesconi, eds, *Frontiers in Artificial Intelligence and Applications*, Vol. 188, IOS Press, 2009, pp. 165–175. doi:10.3233/978-1-58603-942-4-165.
- [7] Y. Hu, S. Ghosh, T. Nguyen and S. Razniewski, GPTKB: Building Very Large Knowledge Bases from Language Models, *CoRR* **abs/2411.04920** (2024). doi:10.48550/ARXIV.2411.04920. <https://doi.org/10.48550/arXiv.2411.04920>.
- [8] X. Wei, X. Cui, N. Cheng, X. Wang, X. Zhang, S. Huang, P. Xie, J. Xu, Y. Chen, M. Zhang, Y. Jiang and W. Han, Zero-Shot Information Extraction via Chatting with ChatGPT, *CoRR* **abs/2302.10205** (2023). doi:10.48550/ARXIV.2302.10205. <https://doi.org/10.48550/arXiv.2302.10205>.
- [9] H. Babaei Giglou, J. D’Souza and S. Auer, LLMs4OL: Large language models for ontology learning, in: *International Semantic Web Conference*, Springer, 2023, pp. 408–427.
- [10] R. Alharbi, V. Tamma, F. Grasso and T.R. Payne, Investigating Open Source LLMs to Retrofit Competency Questions in Ontology Engineering, in: *Proceedings of the AAI Symposium Series*, Vol. 4, 2024, pp. 188–198.
- [11] R. Alharbi, V. Tamma, F. Grasso and T.R. Payne, The role of Generative AI in competency question retrofitting, in: *European Semantic Web Conference*, Springer, 2024, pp. 3–13.
- [12] Y. Zhao, N. Vetter and K. Aryan, Using large language models for ontoclean-based ontology refinement, *arXiv preprint arXiv:2403.15864* (2024).
- [13] S.S. Norouzi, A. Barua, A. Christou, N. Gautam, A. Eells, P. Hitzler and C. Shimizu, Ontology population using LLMs, in: *Handbook on Neurosymbolic AI and Knowledge Graphs*, IOS Press, 2025, pp. 421–438.
- [14] R. Amini, S.S. Norouzi, P. Hitzler and R. Amini, Towards complex ontology alignment using large language models, in: *International Knowledge Graph and Semantic Web Conference*, Springer, 2024, pp. 17–31.
- [15] 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*, 2025.
- [16] N. Fathallah, A. Das, S.D. 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 - Hersonissos, Crete, Greece, May 26-30, 2024, Proceedings, Part I*, A. Meroño-Peñuela, Ó. Corcho, P. Groth, E. Simperl, V. Tamma, A.G. Nuzzolese, M. Poveda-Villalón, M. Sabou, V. Prestiti, I. Celino, A. Revenko, J. Raad, B. Sartini and P. Lisena, eds, *Lecture Notes in Computer Science*, Vol. 15344, Springer, 2024, pp. 36–50. doi:10.1007/978-3-031-78952-6_4.
- [17] N. Fathallah, S. Staab and A. Algergawy, LLMs4Life: Large Language Models for Ontology Learning in Life Sciences, in: *Proceedings of the ELMKE Workshop on Evaluation of Language Models in Knowledge Engineering, EKAW-24 (24th International Conference on Knowledge Engineering and Knowledge Management)*, 2024. <https://arxiv.org/abs/2412.02035>.
- [18] L.M.V. da Silva, A. Kocher, F. Gehlhoff and A. Fay, On the use of large language models to generate capability ontologies, in: *2024 IEEE 29th International Conference on Emerging Technologies and Factory Automation (ETFA)*, IEEE, 2024, pp. 1–8.
- [19] 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 . . . , 2001.
- [20] W. Wong, W. Liu and M. Bennamoun, Ontology learning from text: A look back and into the future, *ACM Comput. Surv.* **44**(4) (2012), 20:1–20:36. doi:10.1145/2333112.2333115.
- [21] C. Biemann, Ontology Learning from Text: A Survey of Methods, *LDV Forum* **20**(2) (2005), 75–93. http://www.jlcl.org/2005_Heft2/Chris_Biemann.pdf.
- [22] M. Hearst, *Automated Discovery of WordNet Relations.* Wordnet An Electronic Lexical Database, MIT Press, Cambridge, MA, 1998.
- [23] N. Aussenac-Gilles, B. Biebow and S. Szulman, Revisiting Ontology Design: A Methodology Based on Corpus Analysis, in: *Knowledge Acquisition, Modeling and Management, 12th International Conference, EKAW 2000, Juan-les-Pins, France, October 2-6, 2000, Proceedings*, R. Dieng and O. Corby, eds, *Lecture Notes in Computer Science*, Vol. 1937, Springer, 2000, pp. 172–188. doi:10.1007/3-540-39967-4_13.
- [24] A. Maedche and S. Staab, Ontology Learning for the Semantic Web, *IEEE Intelligent Systems* **16**(2) (2001), 72–79. doi:10.1109/5254.920602.

- [25] P. Cimiano and J. Völker, Text2Onto, in: *Natural Language Processing and Information Systems, 10th International Conference on Applications of Natural Language to Information Systems, NLDB 2005, Alicante, Spain, June 15-17, 2005, Proceedings*, A. Montoyo, R. Muñoz and E. Métais, eds, Lecture Notes in Computer Science, Vol. 3513, Springer, 2005, pp. 227–238. doi:10.1007/11428817_21.
- [26] F.N. Al-Aswadi, C.H. Yong and K.H. Gan, Automatic ontology construction from text: a review from shallow to deep learning trend, *Artif. Intell. Rev.* **53**(6) (2020), 3901–3928. doi:10.1007/S10462-019-09782-9. <https://doi.org/10.1007/s10462-019-09782-9>.
- [27] R. Lourdasamy and S. Abraham, A survey on methods of ontology learning from text, in: *International Conference on Information, Communication and Computing Technology*, Springer, 2019, pp. 113–123.
- [28] Ó. Corcho, M. Fernández-López and A. Gómez-Pérez, Ontological Engineering: Principles, Methods, Tools and Languages, in: *Ontologies for Software Engineering and Software Technology*, C. Calero, F. Ruiz and M. Piattini, eds, Springer, 2006, pp. 1–48. doi:10.1007/3-540-34518-3_1.
- [29] M. Fernández-López, A. Gómez-Pérez and N. Juristo Juzgado, Methontology: from ontological art towards ontological engineering (1997).
- [30] H.S. Pinto, S. Staab and C. Tempich, DILIGENT: Towards a fine-grained methodology for Distributed, Loosely-controlled and evolving Engineering of oNTologies, in: *Proceedings of the 16th European Conference on Artificial Intelligence, ECAI'2004, including Prestigious Applicants of Intelligent Systems, PAIS 2004, Valencia, Spain, August 22-27, 2004*, R.L. de Mántaras and L. Saitta, eds, IOS Press, 2004, pp. 393–397.
- [31] S. Peroni, A Simplified Agile Methodology for Ontology Development, in: *OWL: - Experiences and Directions - Reasoner Evaluation - 13th International Workshop, OWLED 2016, and 5th International Workshop, ORE 2016, Bologna, Italy, November 20, 2016, Revised Selected Papers*, M. Dragoni, M. Poveda-Villalón and E. Jiménez-Ruiz, eds, Lecture Notes in Computer Science, Vol. 10161, Springer, 2016, pp. 55–69. doi:10.1007/978-3-319-54627-8_5.
- [32] S. Auer, The RapidOWL Methodology—Towards Agile Knowledge Engineering, in: *15th IEEE International Workshops on Enabling Technologies: Infrastructures for Collaborative Enterprises (WETICE 2006), 26-28 June 2006, Manchester, United Kingdom*, IEEE Computer Society, 2006, pp. 352–357. doi:10.1109/WETICE.2006.67.
- [33] V. Presutti, E. Daga, A. Gangemi and E. Blomqvist, eXtreme Design with Content Ontology Design Patterns, in: *Proceedings of the Workshop on Ontology Patterns (WOP 2009), collocated with the 8th International Semantic Web Conference (ISWC-2009), Washington D.C., USA, 25 October, 2009*, E. Blomqvist, K. Sandkuhl, F. Scharffe and V. Svátek, eds, CEUR Workshop Proceedings, Vol. 516, CEUR-WS.org, 2009. <https://ceur-ws.org/Vol-516/pap21.pdf>.
- [34] P. Haase, H. Lewen, R. Studer, D.T. Tran, M. Erdmann, M. d’Aquin and E. Motta, The neon ontology engineering toolkit, *World Wide Web Journal (WWW)* (2008).
- [35] M.C. Suárez-Figueroa, A. Gómez-Pérez and M. Fernández-López, The NeOn Methodology framework: A scenario-based methodology for ontology development, *Applied Ontology* **10**(2) (2015), 107–145. doi:10.3233/AO-150145.
- [36] M. Poveda-Villalón, A. Fernández-Izquierdo, M. Fernández-López and R. García-Castro, LOT: An industrial oriented ontology engineering framework, *Eng. Appl. Artif. Intell.* **111** (2022), 104755. doi:10.1016/J.ENGAPPAL.2022.104755. <https://doi.org/10.1016/j.engappai.2022.104755>.
- [37] A. Sattar, E.S.M. Surin, M.N. Ahmad, M. Ahmad and A.K. Mahmood, Comparative analysis of methodologies for domain ontology development: A systematic review, *International Journal of Advanced Computer Science and Applications* **11**(5) (2020).
- [38] B. Stadlhofer, P. Salhofer and A. Durlacher, An overview of ontology engineering methodologies in the context of public administration (2012).
- [39] V.K. Kommineni, B. König-Ries and S. Samuel, Towards the Automation of Knowledge Graph Construction using Large Language Models **3874** (2024), 19–34. <https://ceur-ws.org/Vol-3874/paper2.pdf>.
- [40] B.P. Allen, L. Stork and P. Groth, Knowledge Engineering Using Large Language Models, *TGDK* **1**(1) (2023), 3:1–3:19. doi:10.4230/TGDK.1.1.3.
- [41] T. Xu, Y. Gu, M. Xue, R. Gu, B. Li and X. Gu, Knowledge graph construction for heart failure using large language models with prompt engineering, *Frontiers Comput. Neurosci.* **18** (2024). doi:10.3389/FNCOM.2024.1389475. <https://doi.org/10.3389/fncom.2024.1389475>.
- [42] R. Alharbi, U. Ahmed, D. Dobryi, W. Łajewska, L. Menotti, M.J. Saeedizade and M. Dumontier, Ex-

ploring the role of generative AI in constructing knowledge graphs for drug indications with medical context, *Proceedings* <http://ceur-ws.org> ISSN **1613** (2023), 0073.

- [43] P. Mateiu and A. Groza, Ontology engineering with Large Language Models, in: *25th International Symposium on Symbolic and Numeric Algorithms for Scientific Computing, SYNASC 2023, Nancy, France, September 11-14, 2023*, IEEE, 2023, pp. 226–229. doi:10.1109/SYNASC61333.2023.00038.
- [44] H.B. Giglou, J. D’Souza and S. Auer, LLMs4OL: Large Language Models for Ontology Learning, in: *The Semantic Web - ISWC 2023 - 22nd International Semantic Web Conference, Athens, Greece, November 6-10, 2023, Proceedings, Part I*, Lecture Notes in Computer Science, Vol. 14265, Springer, 2023, pp. 408–427. doi:10.1007/978-3-031-47240-4_22.
- [45] V.K. Kommineni, B. König-Ries and S. Samuel, From human experts to machines: An LLM supported approach to ontology and knowledge graph construction, *CoRR* **abs/2403.08345** (2024). doi:10.48550/ARXIV.2403.08345. <https://doi.org/10.48550/arXiv.2403.08345>.
- [46] M.J. Saeedizade and E. Blomqvist, Navigating Ontology Development with Large Language Models, in: *The Semantic Web - 21st International Conference, ESWC 2024, Hersonissos, Crete, Greece, May 26-30, 2024, Proceedings, Part I*, Lecture Notes in Computer Science, Vol. 14664, Springer, 2024, pp. 143–161. doi:10.1007/978-3-031-60626-7_8.
- [47] B. Zhang, V.A. Carriero, K. Schreiberhuber, S. Tsaneva, L.S. González, J. Kim and J. de Berardinis, OntoChat: a Framework for Conversational Ontology Engineering using Language Models, *CoRR* **abs/2403.05921** (2024). doi:10.48550/ARXIV.2403.05921. <https://doi.org/10.48550/arXiv.2403.05921>.
- [48] A.S. Lippolis, M. Ceriani, S. Zuppiroli and A.G. Nuzzolese, Ontogenia: Ontology Generation with Metacognitive Prompting in Large Language Models, in: *The Semantic Web: ESWC 2024 Satellite Events - Hersonissos, Crete, Greece, May 26-30, 2024, Proceedings, Part I*, A. Meroño-Peñuela, Ó. Corcho, P. Groth, E. Simperl, V. Tamma, A.G. Nuzzolese, M. Poveda-Villalón, M. Sabou, V. Presutti, I. Celino, A. Revenko, J. Raad, B. Sartini and P. Lisena, eds, Lecture Notes in Computer Science, Vol. 15344, Springer, 2024, pp. 259–265. doi:10.1007/978-3-031-78952-6_38.
- [49] D. Doumanas, A. Soularidis, D. Spiliotopoulos, C. Vassilakis and K. Kotis, Fine-Tuning Large Language Models for Ontology Engineering: A Comparative Analysis of GPT-4 and Mistral, *Applied Sciences* **15**(4) (2025), 2146.
- [50] L. Aung, A. Eberhart, P. Haase, N. Heist, L. Kovriguina, D. Lamprecht and N. Mashhaditafreshi, metis: AI Agent Platform for Human-AI Interaction with Knowledge Graphs, in: *International Semantic Web Conference, ISWC 2025, Nara, Japan, November 2-6, 2025*, 2025.
- [51] J. White, Q. Fu, S. Hays, M. Sandborn, C. Olea, H. Gilbert, A. Elnashar, J. Spencer-Smith and D.C. Schmidt, A Prompt Pattern Catalog to Enhance Prompt Engineering with ChatGPT, *CoRR* **abs/2302.11382** (2023). doi:10.48550/ARXIV.2302.11382. <https://doi.org/10.48550/arXiv.2302.11382>.
- [52] P. Sahoo, A.K. Singh, S. Saha, V. Jain, S. Mondal and A. Chadha, A Systematic Survey of Prompt Engineering in Large Language Models: Techniques and Applications, *CoRR* **abs/2402.07927** (2024). doi:10.48550/ARXIV.2402.07927. <https://doi.org/10.48550/arXiv.2402.07927>.
- [53] T.B. Brown, B. Mann, N. Ryder, M. Subbiah, J. Kaplan, P. Dhariwal, A. Neelakantan, P. Shyam, G. Sastry, A. Askell, S. Agarwal, A. Herbert-Voss, G. Krueger, T. Henighan, R. Child, A. Ramesh, D.M. Ziegler, J. Wu, C. Winter, C. Hesse, M. Chen, E. Sigler, M. Litwin, S. Gray, B. Chess, J. Clark, C. Berner, S. McCandlish, A. Radford, I. Sutskever and D. Amodei, Language Models are Few-Shot Learners, in: *Advances in Neural Information Processing Systems 33: Annual Conference on Neural Information Processing Systems 2020, NeurIPS 2020, December 6-12, 2020, virtual*, H. Larochelle, M. Ranzato, R. Hadsell, M. Balcan and H. Lin, eds, 2020. <https://proceedings.neurips.cc/paper/2020/hash/1457c0d6bfc4967418bfb8ac142f64a-Abstract.html>.
- [54] C.H. Song, B.M. Sadler, J. Wu, W. Chao, C. Washington and Y. Su, LLM-Planner: Few-Shot Grounded Planning for Embodied Agents with Large Language Models, in: *IEEE/CVF International Conference on Computer Vision, ICCV 2023, Paris, France, October 1-6, 2023*, IEEE, 2023, pp. 2986–2997. doi:10.1109/ICCV51070.2023.00280.
- [55] H. Ye, N. Zhang, S. Deng, X. Chen, H. Chen, F. Xiong, X. Chen and H. Chen, Ontology-enhanced Prompt-tuning for Few-shot Learning, in: *WWW ’22: The ACM Web Conference 2022, Virtual Event, Lyon, France, April 25 - 29, 2022*, F. Laforest, R. Troncy, E. Simperl, D. Agarwal, A. Gionis, I. Herman and L. Médini, eds, ACM, 2022, pp. 778–787. doi:10.1145/3485447.3511921.
- [56] A. Kong, S. Zhao, H. Chen, Q. Li, Y. Qin, R. Sun, X. Zhou, E. Wang and X. Dong, Bet-

- ter Zero-Shot Reasoning with Role-Play Prompting, in: *Proceedings of the 2024 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies (Volume 1: Long Papers)*, NAACL 2024, Mexico City, Mexico, June 16-21, 2024, K. Duh, H. Gómez-Adorno and S. Bethard, eds, Association for Computational Linguistics, 2024, pp. 4099–4113. doi:10.18653/V1/2024.NAACL-LONG.228. <https://doi.org/10.18653/v1/2024.naacl-long.228>.
- [57] A. Das, N.S. Fathallah and N. Obretincheva, Navigating Nulls, Numbers and Numerous Entities: Robust Knowledge Base Construction from Large Language Models, in: *KBC-LM/LM-KBC@ ISWC*, 2024.
- [58] J. Wei, X. Wang, D. Schuurmans, M. Bosma, B. Ichter, F. Xia, E.H. Chi, Q.V. Le and D. Zhou, Chain-of-Thought Prompting Elicits Reasoning in Large Language Models, in: *Advances in Neural Information Processing Systems 35: Annual Conference on Neural Information Processing Systems 2022, NeurIPS 2022, New Orleans, LA, USA, November 28 - December 9, 2022*, S. Koyejo, S. Mohamed, A. Agarwal, D. Belgrave, K. Cho and A. Oh, eds, 2022. http://papers.nips.cc/paper_files/paper/2022/hash/9d5609613524ecf4f15af0f7b31abca4-Abstract-Conference.html.
- [59] F. Polat, I. Tiddi and P. Groth, Testing prompt engineering methods for knowledge extraction from text, *Semantic Web* **16**(2) (2025), SW-243719.
- [60] S. Krishna, C. Agarwal and H. Lakkaraju, Understanding the Effects of Iterative Prompting on Truthfulness, in: *Forty-first International Conference on Machine Learning, ICML 2024, Vienna, Austria, July 21-27, 2024*, OpenReview.net, 2024. <https://openreview.net/forum?id=KjazcKPMME>.
- [61] B. Peng, M. Galley, P. He, H. Cheng, Y. Xie, Y. Hu, Q. Huang, L. Liden, Z. Yu, W. Chen and J. Gao, Check Your Facts and Try Again: Improving Large Language Models with External Knowledge and Automated Feedback, *CoRR* **abs/2302.12813** (2023). doi:10.48550/ARXIV.2302.12813. <https://doi.org/10.48550/arXiv.2302.12813>.
- [62] D. Krech, G.A. Grimnes, G. Higgins, J. Hees, I. Aucamp, N. Lindström, N. Arndt, A. Sommer, E. Chuc, I. Herman, A. Nelson, J. McCusker, T. Gillespie, T. Kluyver, F. Ludwig, P.-A. Champin, M. Watts, U. Holzer, E. Summers, W. Morriss, D. Winston, D. Perttula, F. Kovacevic, R. Chateaneu, H. Solbrig, B. Cogrel and V. Stuart, RDFLib, 2023. doi:10.5281/zenodo.6845245. <https://github.com/RDFLib/rdfLib>.
- [63] B. Glimm, I. Horrocks, B. Motik, G. Stoilos and Z. Wang, Hermit: an OWL 2 reasoner, *Journal of automated reasoning* **53** (2014), 245–269.
- [64] E. Sirin, B. Parsia, B.C. Grau, A. Kalyanpur and Y. Katz, Pellet: A practical OWL-DL reasoner, *J. Web Semant.* **5**(2) (2007), 51–53. doi:10.1016/J.WEBSEM.2007.03.004. <https://doi.org/10.1016/j.websem.2007.03.004>.
- [65] 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. <https://doi.org/10.4018/ijswis.2014040102>.
- [66] O.E.G. (OEG), OOPS! - Ontology Pitfall Scanner, 2024, Accessed: April 2025.
- [67] A. Maedche and S. Staab, Measuring similarity between ontologies, in: *International Conference on Knowledge Engineering and Knowledge Management*, Springer, 2002, pp. 251–263.
- [68] S.P. Ponzetto, M. Strube et al., Deriving a large scale taxonomy from Wikipedia, in: *AAAI*, Vol. 7, 2007, pp. 1440–1445.
- [69] E. Zavitsanos, G. Paliouras and G.A. Vouros, Gold standard evaluation of ontology learning methods through ontology transformation and alignment, *IEEE Transactions on knowledge and data engineering* **23**(11) (2010), 1635–1648.
- [70] M.A. Musen, The protégé project: a look back and a look forward, *AI Matters* **1**(4) (2015), 4–12. doi:10.1145/2757001.2757003.
- [71] A. Gangemi, C. Catenacci, M. Ciaramita and J. Lehmann, Modelling ontology evaluation and validation, in: *European semantic web conference*, Springer, 2006, pp. 140–154.
- [72] M. Fernández, C. Overbeeke, M. Sabou and E. Motta, What makes a good ontology? A case-study in fine-grained knowledge reuse, in: *Asian semantic web conference*, Springer, 2009, pp. 61–75.
- [73] L.-I. Wu and G. Li, Zero-shot construction of Chinese medical knowledge graph with ChatGPT, in: *2023 IEEE International Conference on Medical Artificial Intelligence (MedAI)*, IEEE, 2023, pp. 278–283.
- [74] R.M. Bakker, D.L. Di Scala and M. de Boer, Ontology learning from text: an analysis on llm performance, in: *Proceedings of the 3rd NLP4KGC International Workshop on Natural Language Processing for Knowledge Graph Creation, colocated with Semantics*, 2024, pp. 17–19.

- [75] M.A. Cappelli and G. Di Marzo Serugendo, Methodological exploration of ontology generation with a dedicated large language model, *Electronics* **14**(14) (2025), 2863.
- [76] A.S. Lippolis, M.J. Saeedizade, R. Keski-Särkkä, S. Zuppiroli, M. Ceriani, A. Gangemi, E. Blomqvist and A.G. Nuzzolese, Ontology generation using large language models, in: *European Semantic Web Conference*, Springer, 2025, pp. 321–341.
- [77] M. Cheatham and P. Hitzler, String similarity metrics for ontology alignment, in: *International semantic web conference*, Springer, 2013, pp. 294–309.
- [78] J. Li, J. Tang, Y. Li and Q. Luo, Rimom: A dynamic multistrategy ontology alignment framework, *IEEE Transactions on Knowledge and data Engineering* **21**(8) (2008), 1218–1232.
- [79] J. Raad and C. Cruz, A survey on ontology evaluation methods, in: *International conference on knowledge engineering and ontology development*, Vol. 2, SciTePress, 2015, pp. 179–186.
- [80] K. Dellschaft and S. Staab, On how to perform a gold standard based evaluation of ontology learning, in: *International semantic web conference*, Springer, 2006, pp. 228–241.
- [81] J. Plu, O.M. Escobar, E. Trouillez, A. Gapin and R. Troncy, A comprehensive benchmark for evaluating llm-generated ontologies, in: *The Semantic Web-ISWC*, 2024.
- [82] V. Arsenyan, S. Bughdaryan, F. Shaya, K.W. Small and D. Shahnazaryan, Large language models for biomedical knowledge graph construction: information extraction from EMR notes, in: *Proceedings of the 23rd Workshop on Biomedical Natural Language Processing*, 2024, pp. 295–317.
- [83] M. Kulmanov, F.Z. Smaili, X. Gao and R. Hoehndorf, Semantic similarity and machine learning with ontologies, *Briefings in bioinformatics* **22**(4) (2021), bbaa199.
- [84] D. Gromann and T. Declerck, Comparing pretrained multilingual word embeddings on an ontology alignment task, in: *Proceedings of the eleventh international conference on language resources and evaluation (LREC 2018)*, 2018.
- [85] I. Nkisi-Orji, N. Wiratunga, S. Massie, K.-Y. Hui and R. Heaven, Ontology alignment based on word embedding and random forest classification, in: *Joint European Conference on Machine Learning and Knowledge Discovery in Databases*, Springer, 2018, pp. 557–572.
- [86] Z. Hao, W. Mayer, J. Xia, G. Li, L. Qin and Z. Feng, Ontology alignment with semantic and structural embeddings, *Journal of Web Semantics* **78** (2023), 100798.
- [87] A. Khalov and O. Ataeva, Automating Ontology Mapping in IT Service Management: A DOLCE and ITSMO Integration, *Data Science Journal* **24** (2025).
- [88] P. Devkota, S.D. Mohanty and P. Manda, Improving the evaluation of nlp approaches for scientific text annotation with ontology embedding-based semantic similarity metrics, in: *Proceedings of the 20th International Conference on Natural Language Processing (ICON)*, 2023, pp. 516–522.
- [89] A. Bhatt, N. Vaghela and K. Dudhia, Generating Knowledge Graphs from Large Language Models: A Comparative Study of GPT-4, LLaMA 2, and BERT, *arXiv preprint arXiv:2412.07412* (2024).
- [90] F. Friendly, Jaro–Winkler distance improvement for approximate string search using indexing data for multiuser application, in: *Journal of Physics: Conference Series*, Vol. 1361, IOP Publishing, 2019, p. 012080.
- [91] S.J. Grannis, J.M. Overhage and C. McDonald, Real world performance of approximate string comparators for use in patient matching, in: *MEDINFO 2004*, IOS Press, 2004, pp. 43–47.
- [92] A. Karakasidis and E. Pitoura, Identifying Bias in Name Matching Tasks., in: *EDBT*, 2019, pp. 626–629.
- [93] U. Draibach and F. Naumann, On choosing thresholds for duplicate detection., in: *ICIQ*, 2013.
- [94] G. Sousa, R. Lima and C. Trojahn, Results of PropMatch in OAEI 2023., in: *OM@ ISWC*, 2023, pp. 178–183.
- [95] N. Reimers and I. Gurevych, Sentence-BERT: Sentence Embeddings using Siamese BERT-Networks, in: *Proceedings of the 2019 Conference on Empirical Methods in Natural Language Processing and the 9th International Joint Conference on Natural Language Processing (EMNLP-IJCNLP)*, Association for Computational Linguistics, 2019, p. 3982.
- [96] N. Muennighoff, N. Tazi, L. Magne and N. Reimers, Mteb: Massive text embedding benchmark, in: *Proceedings of the 17th Conference of the European Chapter of the Association for Computational Linguistics*, 2023, pp. 2014–2037.
- [97] L. Meyer, C. Stadler, J. Frey, N. Radtke, K. Junghanns, R. Meissner, G. Dziwis, K. Bulert and M. Martin, LLM-assisted Knowledge Graph Engineering: Experiments with ChatGPT, in: *First Working Conference on Artificial Intelligence Development for a Resilient and Sustainable Tomorrow - AI Tomorrow 2023, Leipzig, Germany, 29-30 June, 2023*, C. Zinke-Wehlmann and J. Friedrich, eds, Informatik Aktuell, Springer, 2023, pp. 103–115. doi:10.1007/978-3-658-43705-3_8.

Appendix

A. Prompt Templates

A.1. Domain-specific Persona Generation Prompt Template

Domain-specific Persona Generation Prompt Template

You are an expert in knowledge and ontology engineering, following best practices from the NeOn methodology and foundational ontology engineering approaches. Your task is to assume a professional persona tailored to a specific domain to guide ontology design, reuse, and population.

Instructions:

1. Based on the provided domain name and description, define a detailed professional persona that:
 - Reflects knowledge of standards, tools, datasets, and ontology reuse
 - Is realistic and usable as a context for prompting LLMs
2. The persona should reflect familiarity with domain-specific knowledge and semantic web technologies (OWL, RDF, SPARQL, Turtle), modular design, and reuse of existing vocabularies and patterns.

- Domain Name: {domain_name}
- Domain Description: {domain_description}

--- Output ---

```
persona = "You are an expert knowledge and ontology engineer  
specializing in {domain_name}..."
```

A.2. Ontology Draft Generation Prompt Templates

1. Requirement Specification

You are a {persona}. The {domain_name} domain is described as follows: {domain_description}.

Use the following keywords to guide your reasoning: {keywords}.

Ground the requirement specification fully in both the domain description and the keywords.

The NeOn methodology starts by specifying the following ontology requirements:

- The purpose of the ontology
- The scope of the ontology
- The target group of the ontology
- The intended uses
- The functional requirements
- The non-functional requirements

--- Output ---

Ontology Requirements:

2. Competency Questions Generation

Based on the Ontology Requirements developed {ontology_requirements}, write a list of Competency Questions that the core module of the ontology should be able to answer. Examples of Competency Questions for reference are provided here:{few_shot_competency_questions}.

--- Output ---
Competency Questions:

3. Conceptual Modeling: Entity and Property Generation

For each Competency Question: {competency_questions}, extract entities and relations (properties) that must be introduced in the {domain_name} ontology. A single competency question can help in extracting more than one triple.

{few_shot_entity_extraction}
{few_shot_property_extraction}

--- Output ---
Entities and Properties:

3.1 Conceptual Modeling: Axiom Generation

Considering all the generated entities, relations (properties):
{generated_entities_properties}

Generate a conceptual model expressing the triples of the entities, properties (relations), and axioms in the format:
(subject-relation-object triples).

- Make sure that your conceptual model is logically consistent and free from common pitfalls (e.g., Wrong inverse relationships, Cycles in a class hierarchy, multiple domains or ranges, and wrong transitive relationships).

--- Output ---
Conceptual Model:

3.2 Conceptual Modeling: Hierarchy Building

Using the conceptual model generated in the previous step: {conceptual_model}, refine and organize the ontology into a coherent class hierarchy.

- Introduce "rdfs:subClassOf" structures that reflect the relationships implied by the existing entities, relations, and axioms.
- Represent all hierarchical assertions using (subject-relation-object) triples, which will be used to extend the existing {conceptual_model}.
- Make sure that the resulting hierarchy is logically consistent and avoids common modelling pitfalls (e.g., cycles in the class hierarchy).

--- Output ---
Hierarchy Triples:

4. Reuse

For the following conceptual model: {conceptual_model}.
Your task is to improve this conceptual model by generating new triples that strengthen its ontology structure.
Enhance the model by reusing knowledge from the example ontology fragments below.

Reuse refers to utilizing existing ontological knowledge or structures as input in the development of new ontologies. It enables more efficient and consistent knowledge representation while improving interoperability across systems and applications.

Reuse the following example to improve the ontology structure from the {reuse_ontology_name_description}.

Example Triples: {few_shot_reuse}

- Print ONLY the new triples you add to the conceptual model.
 - Each triple must follow the format: subject-relation-object.
 - Do NOT repeat or paraphrase any triples from the input conceptual model.
 - Make sure that your conceptual model is logically consistent and free from common pitfalls (e.g., Wrong inverse relationships, Cycles in a class hierarchy, multiple domains or ranges, and wrong transitive relationships).
- Output ---
New Triples:

5. Implementation

Serialize the complete {conceptual_model} developed in the previous step into Turtle syntax.

- Make sure the syntax is correct, all entities and properties have correct prefixes, the ontology is consistent, and free from common pitfalls (e.g., wrong inverse relationships, Cycles in a class hierarchy, multiple domains or ranges, and wrong transitive relationships).
- Use consistent prefixes and namespaces (e.g., : for the ontology namespace).
- Include rdfs:subClassOf and rdf:type statements where appropriate.
- Make sure to Print Turtle code between `###start_turtle###` and `###end_turtle###` markers only.

--- Output ---
Turtle Ontology:
`###start_turtle###`
.....
`###end_turtle###`

6. Formal Modeling: Data Properties

For all the entities in the {Turtle_ontology} given, introduce Data Properties when meaningful.

Here are some examples: {few_shot_data_properties}.

Follow these instructions:

- Add appropriate data properties for entities.
- Assign correct rdfs:domain & rdfs:range datatypes (xsd:string, xsd:date, etc.).
- Make sure the syntax is correct, all entities and properties have correct prefixes, the ontology is consistent, and free from common pitfalls (e.g., wrong inverse relationships, Cycles in a class hierarchy, multiple domains or ranges, and wrong transitive relationships).
- Print ONLY new triples that do not exist in the given ontology.
- Do NOT repeat or regenerate any existing triples.
- Output strictly between `###start_turtle###` and `###end_turtle###` markers

--- Output ---

New Data-Property Triples (Turtle Syntax):

```
###start_turtle###  
... new triples ...  
###end_turtle###
```

6.1 Formal Modeling: Object Properties

For all the entities in the {Turtle_ontology} given, introduce Object Properties when meaningful.

Here are some examples: {few_shot_object_properties}.

Follow these instructions:

- Add appropriate object properties for entities.
- Make sure the syntax is correct, all entities and properties have correct prefixes, the ontology is consistent, and free from common pitfalls (e.g., wrong inverse relationships, Cycles in a class hierarchy, multiple domains or ranges, and wrong transitive relationships).
- Print ONLY new triples that do not exist in the given ontology.
- Do NOT repeat or regenerate any existing triples.
- Output strictly between `###start_turtle###` and `###end_turtle###` markers

--- Output ---

New Data-Property Triples (Turtle Syntax):

```
###start_turtle###  
... new triples ...  
###end_turtle###
```

6.2 Formal Modeling: Object Properties (Inverse, Reflexive, Symmetric, Transitive, Functional)

For all object properties in the {Turtle_ontology}, extend the ontology by introducing additional object-property axioms where meaningful.

```
{property_type}  
{property_explanation}
```

Follow these instructions:

- Maintain correct rdfs:domain and rdfs:range usage.
- Ensure consistent naming conventions for newly added properties.
- Make sure the syntax is correct, all entities and properties have correct prefixes, the ontology is consistent, and free from common pitfalls (e.g., wrong inverse relationships, Cycles in a class hierarchy, multiple domains or ranges, and wrong transitive relationships).
- Print ONLY new triples that do not exist in the given ontology.
- Do NOT repeat or regenerate existing triples.
- Output strictly between ###start_turtle### and ###end_turtle### markers.

--- Output ---

New Object-Property Triples (Turtle Syntax):

```
###start_turtle###  
... new triples ...  
###end_turtle###
```

7. Population

Populate the given {Turtle_ontology} with meaningful real-world individuals. Here are some examples of individuals: {few_shot_individuals}.

Follow these instructions:

- Add named individuals (instances) of the existing classes.
- Ensure each individual has type declarations and relevant property assertions.
- Make sure the syntax is correct, all entities and properties have correct prefixes, the ontology is consistent, and free from common pitfalls (e.g., wrong inverse relationships, Cycles in a class hierarchy, multiple domains or ranges, and wrong transitive relationships).
- Print ONLY new triples that do not exist in the given ontology.
- Do NOT repeat or regenerate existing triples.
- Output strictly between ###start_turtle### and ###end_turtle### markers.

--- Output ---

New Triples (Turtle Syntax):

```
###start_turtle###  
... new triples ...  
###end_turtle###
```

8. Documentation

If not present in the given {Turtle_ontology}, add metadata triples about:

- ontology IRI
- ontology label
- version information
- natural-language description (rdfs:comment)

Follow these instructions:

- Add ontology-level metadata using appropriate properties (owl:Ontology, rdfs:label, owl:versionInfo, rdfs:comment).
- Make sure the syntax is correct, all entities and properties have correct prefixes, the ontology is consistent, and free from common pitfalls (e.g., wrong inverse relationships, Cycles in a class hierarchy, multiple domains or ranges, and wrong transitive relationships).
- Print ONLY new triples that do not exist in the given ontology.
- Do NOT repeat or regenerate existing triples.
- Output strictly between `###start_turtle###` and `###end_turtle###` markers.

--- Output ---

New Triples (Turtle Syntax):

```
###start_turtle###  
... new triples ...  
###end_turtle###
```

A.3. *Ontology Verification and Resolution Prompt Templates*

Ontology Resolution (Syntax Errors, Logical Inconsistencies, Common Modeling Pitfalls) Prompt Template

You are an expert in RDF and Turtle repair. The following Turtle fragment failed to parse.

Error:

{error_message}

Problematic code:

{problem_block}

Full Ontology:

{Turtle_ontology}

Please fix the error while preserving its intended semantics.

Return ONLY valid Turtle triples between the markers.

--- Output ---

New Triples (Turtle Syntax):

```
###start_turtle###  
... new triples ...  
###end_turtle###
```

B. Detailed Evaluation Results

B.1. Structural Analysis Detailed Evaluation Results

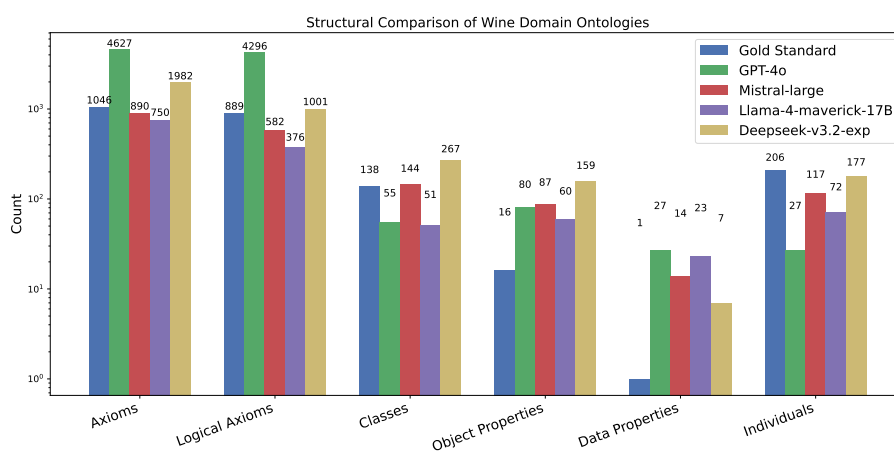


Figure 9. Structural comparison between the NeOn-GPT-generated **Wine Domain ontologies** with 4 LLMs (GPT-4o, Llama4-maverick-17B-128, Mistral, DeepSeek-v3.2-exp) and the gold standard (Stanford Wine). Bars show counts of axioms, logical axioms, classes, properties (object and data types), instances, and subclass relations, highlighting differences in modeled structure and relation density.

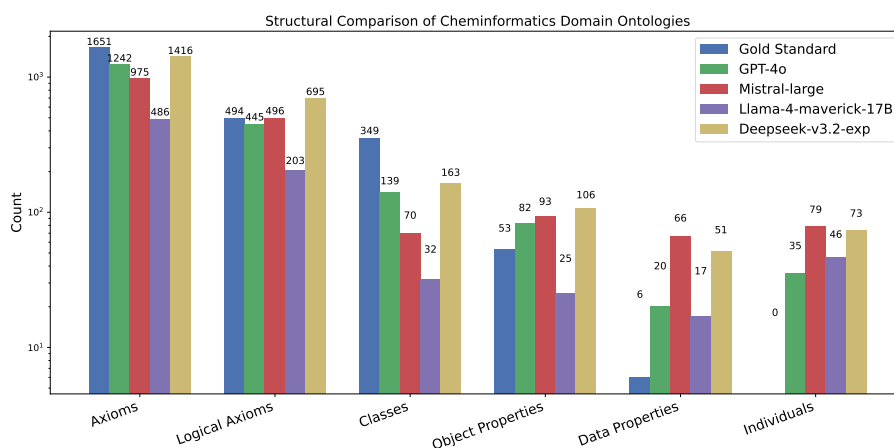


Figure 10. Structural comparison between the NeOn-GPT-generated **Cheminformatics Domain ontologies** with 4 LLMs (GPT-4o, Llama4-maverick-17B-128, Mistral, DeepSeek-v3.2-exp) and the gold standard (CHEMINF). Bars show counts of axioms, logical axioms, classes, properties (object and data types), instances, and subclass relations, highlighting differences in modeled structure and relation density.

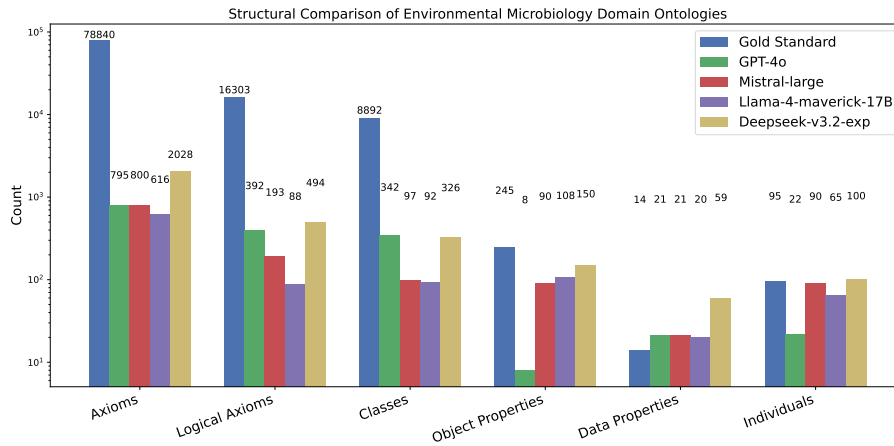


Figure 11. Structural comparison between the NeOn-GPT-generated **Enivormental Microbiology Domain ontologies** with 4 LLMs (GPT-4o, Llama4-maverick-17B-128, Mistral, DeepSeek-v3.2-exp) and the gold standard (AquaDiva). Bars show counts of axioms, logical axioms, classes, properties (object and data types), instances, and subclass relations, highlighting differences in modeled structure and relation density.

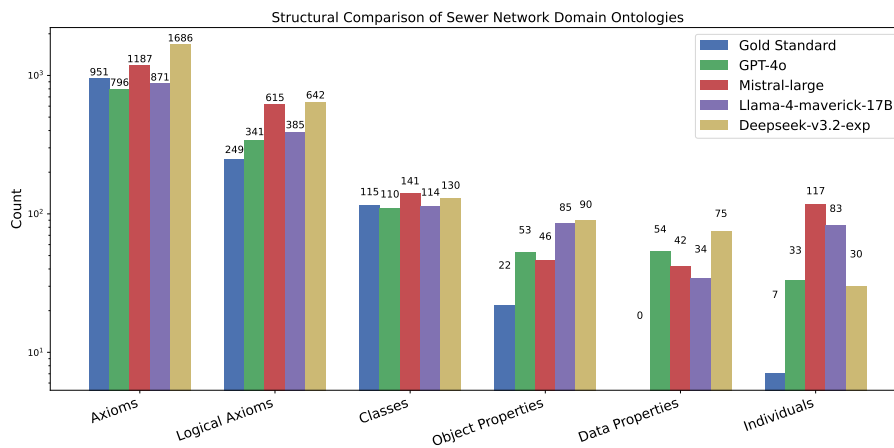


Figure 12. Structural comparison between the NeOn-GPT-generated **Sewer Network Domain ontologies** with 4 LLMs (GPT-4o, Llama4-maverick-17B-128, Mistral, DeepSeek-v3.2-exp) and the gold standard (SewerNet). Bars show counts of axioms, logical axioms, classes, properties (object and data types), instances, and subclass relations, highlighting differences in modeled structure and relation density.

B.2. Lexical Analysis Detailed Evaluation Results

Domain	Ontology	Exact Matched Classes and Properties
Wine	GPT-4o	RedWine, Region, RoseWine, SweetWine, TableWine, WhiteWine, Wine, hasFlavor, hasVintageYear
	Mistral-large	hasvintageyear
	Llama-4-maverick-17B	RedWine, Vintage, WhiteWine, Wine, Winery, hasBody, hasColor, hasFlavor, producesWine,
	Deepseek-v3.2-exp	Bordeaux, Burgundy, CabernetSauvignon, Chardonnay, CheninBlanc, DessertWine, Merlot, PinotNoir, RedWine, Region, Riesling, SauvignonBlanc, Semillon, Vintage, WhiteWine, Wine, WineColor, Winery, Zinfandel, hasBody, hasColor, hasFlavor, locatedIn, madeFromGrape, producesWine
Cheminformatics	GPT-4o	<i>None</i>
	Mistral-large	<i>None</i>
	Llama-4-maverick-17B	<i>None</i>
	Deepseek-v3.2-exp	<i>None</i>
Environmental Microbiology	GPT-4o	affectedby, biodiversity, foundin, groundwater, groundwaterquality, influence, karst, limestone, marinephage, mineral, nitrogentransformation, permeability, relatedto, vegetation, waterflow,
	Mistral-large	affects, archaea, bacteria, biodiversity, clay, climate, diversity, groundwaterquality, habitat, hascode, humidity, influences, karst, limestone, porosity, precipitation, rarespecies, secondarymineral, species, temperature, uses, usesmethod, viruses
	Llama-4-maverick-17B	affects, contains, habitat, hasmeasurement, hasmember, organism, population, process
	Deepseek-v3.2-exp	abundance, affects, aquifer, archaea, ascomycota, bacteria, basalt, biofilm, borehole, cave, class, clay, depth, diversity, domain, drought, family, flooding, foundin, gene, genus, granite, groundwater, habitat, hasmember, hasprecision, hasunit, identifier, influences, limestone, microscope, order, organism, ph, phylum, population, porewater, presentin, pressure, sandstone, shale, species, spring, temperature, temporalentity
Sewer Networks	GPT-4o	inlet, inspection, maintenance, manhole, pipe, pumpingstation, repair, retention-basin
	Mistral-large	inlet, inspection, manhole, pipe, pumpingstation, repair, state, stormwaternet-work, wastewaternetwork
	Llama-4-maverick-17B	geometry, hastime, inlet, inspection, maintenance, manhole, network, pipe, pumpingstation, repair, slope
	Deepseek-v3.2-exp	inlet, inspection, maintenance, manhole, pipe, pumpingstation, repair, stormwa-ternetwork, treatmentplant, valve, wastewaternetwork,

Table 6. Exact lexical matches between gold-standard ontologies and LLM-generated ontologies across four domains (Wine, Cheminformatics, Environmental Microbiology, Sewer Networks). For each model, the table lists all classes and properties whose local names match the corresponding gold entities verbatim.

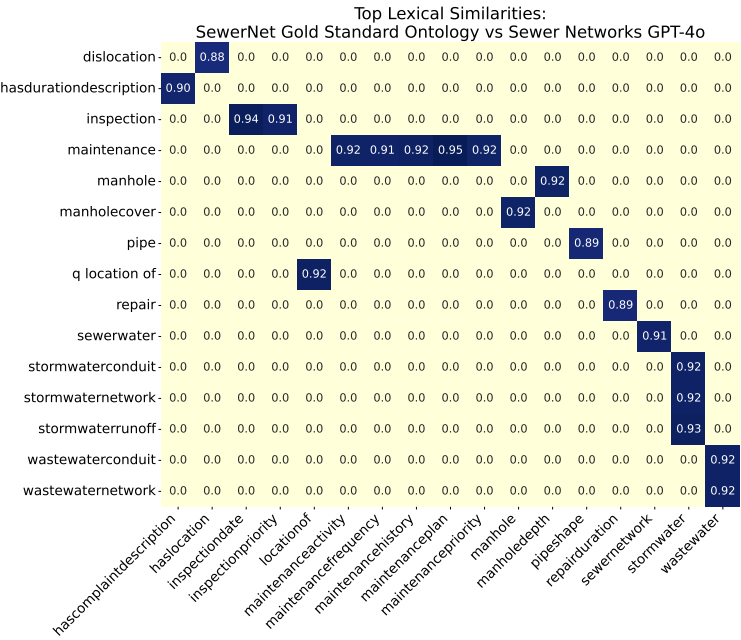


Figure 28. (a) GPT-4o

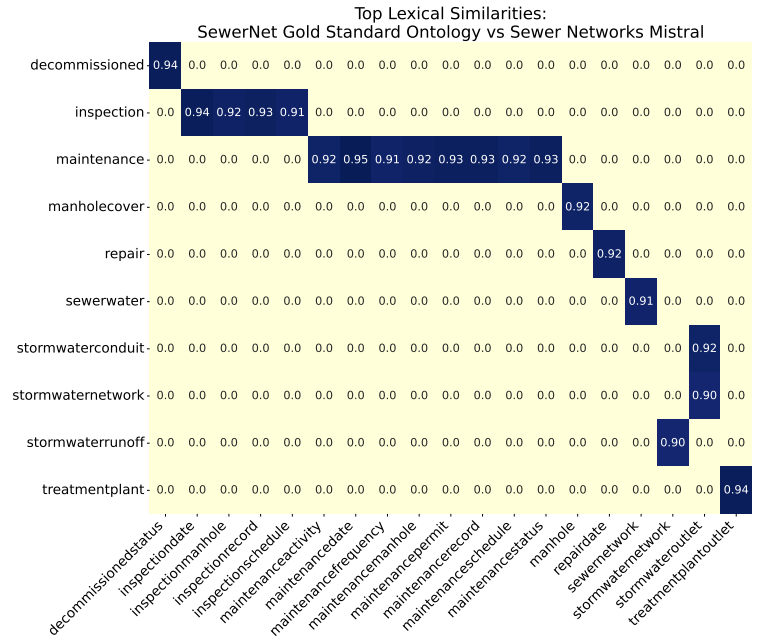


Figure 29. (b) Mistral-large

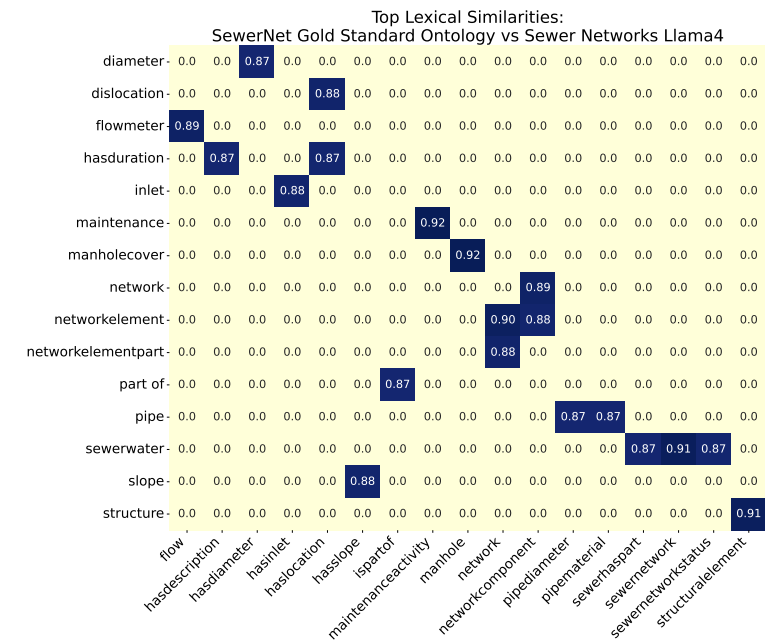


Figure 30. (c) Llama-4

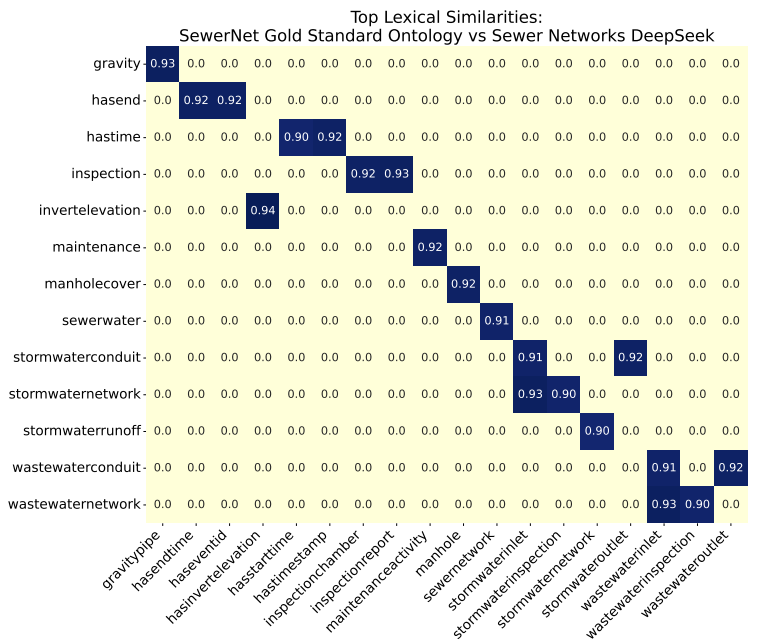


Figure 31. (d) Deepseek

Figure 32. Top lexical similarity heatmap between the **SewerNet Gold Standard Ontology** and the **NeOn-GPT-generated Sewer Networks** ontology using four LLMs: (a) GPT-4o, (b) Mistral-large, (c) Llama-4, and (d) DeepSeek. Exact lexical matches are removed; only the top 20 non-identical similarities are shown. The x-axis lists LLM-generated entities and properties, and the y-axis lists their Gold Standard counterparts. Darker cells denote stronger lexical similarity between the two ontologies.

C. Critical OOPS! Pitfalls Detected in the NeOn-GPT Generated Ontologies

This section provides the definitions of the critical modeling pitfalls reported in Table 5. All definitions correspond to the official OOPS! catalog and are listed in numerical order for reference.

- P05. Defining wrong inverse relationships** Inverse relationships are defined between properties that do not actually represent inverse semantics. This may lead to incorrect inferences when reasoning over object properties.
- P06. Including cycles in a class hierarchy** Cycles are introduced in the class subsumption hierarchy (e.g., a class is declared as a subclass of itself, directly or indirectly), which can lead to unintended logical consequences.
- P19. Defining multiple domains or ranges in properties** A property is defined with multiple domain or range axioms, which in OWL are interpreted conjunctively rather than disjunctively, often resulting in unintended restrictions.
- P27. Defining wrong equivalent properties** Two object properties or datatype properties are declared as equivalent using `owl:equivalentProperty`, even though they do not share the same semantics.
- P28. Defining wrong symmetric relationships** A property is declared as symmetric even though its semantics are not symmetric, leading to incorrect bidirectional inferences.
- P29. Defining wrong transitive relationships** A property is declared as transitive despite not satisfying transitivity semantics, which can propagate incorrect relationships through inference.
- P31. Defining wrong equivalent classes** Classes are declared equivalent using `owl:equivalentClass` even though they do not represent the same conceptual meaning, potentially collapsing distinct concepts.
- P39. Ambiguous namespace** The ontology uses namespaces in an ambiguous or inconsistent manner, which can hinder interpretation, reuse, or integration with other ontologies.