NeOn-GPT: A Large Language Model-Powered Pipeline for Ontology Learning*

Nadeen Fathallah[1], Arunav Das[2], Stefano De Giorgis[3], Andrea Poltronieri[4], Peter Haase[5], and Liubov Kovriguina[5]

[1] Analytic Computing, Institute for Artificial Intelligence, University of Stuttgart, Germany
[2] King's College London, UK
[3] Institute of Cognitive Sciences and Technologies - National Research Council (ISTC-CNR), University of Bologna, Italy
[4] Department of Computer Science and Engineering, University of Bologna, Italy
[5] metaphacts GmbH, Walldorf, Germany

Abstract. We address the task of ontology learning by combining the structured NeOn methodology framework with Large Language Models (LLMs) for translating natural language domain descriptions into Turtle syntax ontologies. The main contribution of the paper is a prompt pipeline tailored for domain-agnostic modeling, exemplified through the application to a domain-specific case study: the wine ontology. The resulting pipeline is used to develop NeOn-GPT, a workflow for automatic ontology modeling, and is integrated into the metaphactory platform. NeOn-GPT leverages the systematic approach of the NeOn methodology and LLMs' generative capabilities to facilitate a more efficient ontology development process. We evaluate the proposed approach by conducting comprehensive evaluations using the Stanford wine ontology as the gold standard. The obtained results show, that LLMs are not fully equipped to perform procedural tasks required for ontology development, and lack the reasoning skills and domain expertise needed. Overall, LLMs require integration into workflow or trajectory tools for continuous knowledge engineering tasks. Nevertheless, LLMs can significantly alleviate the time and expertise needed. Our code base is publicly available for research and development purposes, accessible at: `https://github.com/andreamust/NEON-GPT`.

Keywords: Ontology Modelling · Large Language Models · NeOn Methodology.

1. Introduction

LLMs have revolutionized the landscape of both artificial intelligence research and our daily life [14,30]. They have shown unparalleled capabilities in understanding, generating, and interpreting human language, thus becoming integral

* **Conceptualization:** N.D., A.D, S.D.G, A.P, P.H; **Investigation:** N.D., A.D, S.D.G, A.P, P.H; **Methodology:** S.D.G, A.P, P.H; **Software:** N.F, A.D, S.D.G, A.P; **Validation:** N.F, A.D; **Visualization:** N.F, A.D; **Writing - original draft:** N.F., A.D; **Writing - review & editing:** N.F, A.D, S.D.G., L.K

tools in various domains ranging from automated content creation to complex decision support systems. However, despite their impressive capabilities, they present several fundamental limitations such as: (i) the lack of embodied grounding, informed by real-world sensorial data [18], (ii) the well-known hallucination problem and incomplete information processing [13], and (iii) and the lack of robust fact-checking mechanisms to validate generated content [28]. These challenges have in turn highlighted the need for ongoing research and development to enhance the reliability, accuracy, and overall utility of LLMs in knowledge-intensive applications, paving the way for the integration of LLMs and knowledge graphs (KGs) on one side using KGs as reinforcement learning and on the other side using LLMs to elicit, provide, and correct generic domain requirements and desiderata [23]. This work is focused on the last point, namely the adoption of LLMs for ontology learning [17], intended as the integration of semi-automatic techniques as a support for the cooperative ontology engineering process.

This work aims to generate a plausible first draft of domain-specific ontologies, starting from mere domain descriptions, provided in natural language, to get an actual formalization of the domain knowledge expressed in Turtle syntax. Our purpose is to enhance the efficiency of the ontology learning process. To do so, we re-engineer the NeOn methodology framework [32,11] to be given as an instructory paradigm for generative LLMs [35].
Our contributions can be summarized as follows:

– Prompt pipeline implementation based on the principles of the NeOn methodology, to facilitate ontology learning for domain-specific ontologies.
– Development of the NeOn-GPT workflow, an automated workflow for ontology generation. NeOn-GPT uses the NeOn methodology to guide LLMs through structured ontology-building phases, including concept refinement and axiom elaboration, with features for validation and soundness checks.
– In-depth evaluation of our approach, with the wine ontology as a use case.

The paper is structured as follows: Section 2, an overview of the existing literature. Sections 3 and 4 present our approach, the NeOn-GPT workflow, and the integration with the metaphactory platform. Section 5 is dedicated to the experiments and in Section 6, the results obtained from applying our methodology. Finally, Section 7 offers a conclusion and future research endeavors.

2. Related Work

Relevant work to this contribution includes ontology learning from text, LLMs for conceptual modeling, and the NeOn methodology for ontology development.

Ontology Learning from Text is the task of automatically extracting and generating the components of an ontology from textual data [17]. An early approach [12], utilizes lexico-syntactic patterns for ontology learning, enabling the automated extraction of hierarchical relationships from large text collections and enhancing existing lexical databases like WordNet. [2] used corpus analysis for ontology construction, by extracting relevant concepts and relationships from the linguistic and domain-specific terminology in text corpora. [17] elaborate on integrating Natural Language Processing (NLP) and Semantic Web applications in ontology learning. They utilize NLP techniques to extract concept hierarchies and relationships from unstructured web content to automate the creation of semantic web annotations. [6] propose Text2Onto framework for data-driven ontology learning, utilizes probabilistic algorithms, association rule mining, and clustering techniques for extracting concepts, relationships, and categorizing entities within ontologies from textual data. These works inspire our use of an established ontology development framework to guide LLMs, by demonstrating the effectiveness of structured rule-based methods and NLP methods in extracting and organizing knowledge from text.

LLMs for Conceptual Modelling, The fusion of LLMs with conceptual modeling practices represents a novel approach to understanding and organizing knowledge. Early experiments like the LAMA benchmark [25] evaluate LMs' understanding of factual and commonsense knowledge through 'cloze' prompts, where the model's task is to accurately complete sentences missing words that represent subject-relation-object triples. This showcases LMs' ability to interpret and utilize structured knowledge required for conceptual modeling. Further experiments use ChatGPT for generating entity-relationship diagrams, to illustrate LLMs' capability to directly translate natural language descriptions into structured conceptual models, leveraging structured prompts and a zero-shot learning approach [7]. [33] use LLMs to visualize data interconnections by introducing GraphGPT, they prompt LLMs to transform textual information into the corresponding knowledge graphs. Another approach for knowledge graph generation is proposed by [4], where LLMs are prompted in a bottom-up approach to initially create an element hierarchy and subsequently identify possible relationships between elements. These works are relevant input to our work as they demonstrate LLMs' capability to interpret and structure domain-specific knowledge into conceptual models.

NeOn Methodology for Ontology Development [32,11] is designed to facilitate ontology development through a comprehensive, scenario-based framework. It enables flexibility with alternative development paths and delivers a structured approach through detailed guidelines for processes and activities associated with ontology development. Contrasted with Agile methodologies like the eXtreme Design (XD) [27], NeOn emphasizes detailed planning over Agile's rapid, minimal design approach [31]. This makes NeOn well-suited for projects that require deep conceptualization and systematic methodology, especially in collaborative and iterative ontology development. NeOn offers a generic framework that aligns with software engineering principles for a broad array of ontology development projects, this sets NeOn apart from more specialized methodologies such as SAMOD [24] and RapidOWL [1].

We follow three distinct phases from the NeOn methodology: (i) specification of ontology requirements — defining purpose, scope, and target group; (ii) ontology conceptualization — establishing class hierarchy and structuring concepts; and (iii) ontology implementation in a specific formalism (e.g., Turtle syntax[3]).

3. Methodology

3.1. NeOn-GPT Workflow

Our proposed approach capitalizes on LLMs' ability to interpret and generate natural language to understand domain descriptions and convert them to their corresponding ontologies. We use the NeOn methodology framework, known for its structure and iterative approach to guide ontology development through distinct phases: requirement specification, ontology conceptualization, and ontology implementation, to ensure that the generated ontology is not only logically sound but also aligned with domain requirements. We convert the NeOn methodology framework to a series of prompts to an LLM, specifically GPT-3.5. This synergy is encapsulated in our workflow (NeOn-GPT) illustrated in Fig. 1.

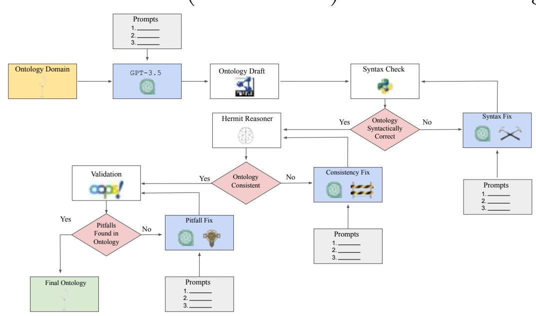


Fig. 1: NeOn-GPT Ontology Engineering Pipeline

Step 1 - Ontology draft generation: Following the NeOn Methodology Framework for ontology development. In particular, we focus on the "Specification of ontology requirements" section, which includes: (i) providing a generic domain description, (ii) specification of the ontology purpose, (iii) scope, and (iv) requirements of the ontology, and (v) example competency questions (CQs). We translate this framework and domain description into a series of structured prompts for an LLM, GPT-3.5 [37]. To enhance prompt quality, we use prompt engineering techniques, such as Chain-of-Thought (CoT) [38] and Role-play prompting [29]. CoT prompts guide GPT-3.5 through a series of logical steps to a final output clarifying LLMs' reasoning process, while Role-play LLMs to adopt specific perspectives or personas (e.g., "Experienced Knowledge Engineer"), resulting in more contextually relevant responses. This approach is detailed in Fig. 2a. We further prompt GPT-3.5 to generate CQs as brief queries that help clarify and assess the knowledge representation within the ontology (see Fig. 2b).

Following this, ontology conceptualization, and conceptual modeling in the NeOn methodology involves entity and relationship extraction. Through few-shot prompting [36], we prompt GPT-3.5 to extract entities and relationships from the generated CQs and generate a conceptual model of the ontology in the form of subject-relation-object triples (see Fig. 3). Few-shot prompting involves the construction of prompts that describe the task, accompanied by a set of examples enabling the LLM to generate contextually relevant outputs. In the context of the wine ontology, examples of entities include `Wine` and

`GrapeVariety`. Examples of properties of the `Wine` entity include `Color` and `SugarLevel`. The domain range for the property `Color` of the `Wine` entity is: `(Red, White, Rosé)`. The relationship between `Wine` and `GrapeVariety` entities is: `Wine hasGrapeVariety GrapeVariety`.

Lastly, for ontology implementation, we prompt GPT-3.5 to use the generated triples to implement a full ontology serialized in Turtle syntax. Following this, we apply formal modeling to accurately capture the domain's complexities and relationships in the ontology, ensuring it is logically sound and capable of supporting advanced reasoning. This entails prompting GPT-3.5 to include object properties: inverse, reflexivity, transitivity, symmetry, and functional. The prompt is constructed such that object properties are added when meaningful while ensuring the ontology remains consistent. An example of the generated triples is shown in Fig. 4. To enhance ontology usability and readability, we prompt GPT-3.5 to enrich entities and relationships with natural language descriptions and add essential metadata such as IRI, labels, and versions (see Fig. 5). To ground the ontology in real-world data and facilitate knowledge discovery, we use few-shot prompting to populate the ontology with real-world instances (see Fig. 6).

Step 2 - Syntax validation: our workflow illustrated in Fig. 1 verifies that the generated ontology draft is syntactically correct using the RDFLib python package [16]. RDFLib detects syntax errors and error messages are used to prompt GPT-3.5 for error correction (example in Fig. 7).

Step 3 - Consistency check: after examining the resulting ontology, it became apparent that introducing domain complexity, also resulted in the introduction of some inconsistencies. To remedy this, we use the HermiT reasoner API [9] to ensure that the resulting ontology is logically sound and free from inconsistencies. The inconsistency error messages generated by the HermiT reasoner are used as prompts to GPT-3.5 for inconsistency resolution, as shown in Fig. 8.

Step 4 - Pitfall resolution: our workflow uses OOPS API [26] for pitfall scanning to validate the generated ontology, common pitfalls include circular axioms and missing disjointness. OOPS API categorizes common pitfalls into three categories: Critical, Important, and Minor. GPT-3.5 showed proficiency in addressing Critical and Important pitfalls but failed to address Minor pitfalls and maintain a consistent and coherent ontology. Validation results from OOPS API are used as prompts for GPT-3.5 for pitfall resolution, as shown in Fig. 9.

A visualization of the final wine ontology structure generated by NeOn-GPT is illustrated in Fig. 10c.

4. NeOn-GPT Integration with metaphactory Platform

An important aspect to facilitate ontology development with LLMs is providing the ontology designer with sufficient guidance for the chosen methodology while ensuring that the output of each step is valid and consistent to qualify as the input for the subsequent step. This poses a challenge in the context of LLMs, as they struggle with procedural tasks, producing hallucinated and fragmented results, or missing the steps of the process.

To address this challenge, we have integrated the NeOn-GPT prompt pipeline into the metaphactory platform [10]. To guide the user consistently through the NeOn methodology, we leveraged the metaphactory built-in workflow ontology, designed for the representation of interactive workflows. This ontology describes a workflow on conceptual and instance levels and allows one to associate instructions, templates, and other artifacts to a particular workflow step (see Fig. 12). For the NeOn-GPT prompt pipeline, the artifacts are prompt templates, prompts, and LLM-generated results. On the definition level, each *Workflow Step* is associated with an ordered set of *Prompt Templates*, which corresponds to the *Workflow States* and its associated *Prompts* and results on the instance level. The ontology implementation step is shown in Fig. 13. The metaphactory built-in workflow ontology enables results from previous steps to inform subsequent ones. The NeOn-GPT metaphactory integration offers a user-friendly interface, acting as a copilot for automated ontology development.

5. Experiments

In our experiments, we use the wine ontology to validate the NeOn-GPT workflow, showcasing its practical utility in ontology learning. The wine ontology [22] is a structured framework that organizes wine attributes and relationships, from grape varieties to taste profiles and production methods, enhancing information systems for wine knowledge management, retrieval, and recommendations.

To identify the suitable LLM for our task, we used GPT-3.5 [37], Llama [8], and PaLm [5] through zero-shot prompting experiments. GPT-3.5 generated syntactically sound and non-redundant ontologies, outperforming Llama and PaLm, thus chosen for developing NeOn-GPT workflow. In our experiments, we prompt LLMs to generate ontologies in Turtle syntax [3] after initial experiments with Owl syntax [19] generated ontologies with syntactical errors. We conduct three experiments to evaluate our approach:(i) Zero-shot Prompting [15], prompt GPT-3.5 with wine domain natural language description to generate the wine ontology in Turtle syntax, (ii) Ontology Development Guide Prompts, parse the Stanford guide to create ontologies [22], to a series of structured prompts for GPT3.5 to generate the wine ontology in Turtle syntax, and (iii) NeOn-GPT Workflow (Our approach).

6. Results

We employed the Protégé Stanford University Wine Ontology [22] as the gold standard ontology for benchmarking our resulting ontologies. Comparing the Zero-Shot Prompting Ontology, Ontology Development Guide Ontology, and NeOn-GPT Ontology against gold standard ontology, illustrated in Fig. 10, revealed that both the Zero-Shot Prompting and Ontology Development Guide ontologies exhibited a flat hierarchy, lacked conceptual structure, and contained syntactical errors and redundant terms. Given NeOn-GPT's wine ontology's superior capacity to capture domain complexity while remaining logically sound and consistent compared to its counterparts, we conduct in-depth evaluations to compare NeOn-GPT's wine ontology to the gold standard wine ontology, illustrated in Table 1. These evaluations encompass both structural assessment and pre-inference and post-inference assessment.

Metric	Gold standard wine ontology	NeOn-GPT wine ontology
Axioms	911	387
Logical axioms count	657	139
Asserted Class count	77	39
Object property count	13	24
Data property count	1	12
Properties count	14	36
Individual count	161	25

Table 1: Ontology Metrics Comparison - gold standard wine ontology and NeOn-GPT wine ontology [source: OntoMetrics [34], Protégé [20]].

6.1. Structural Assessment

Involves schema and hierarchical comparisons. Results from the schema comparison illustrated in Table 1, can be summarized as:

- NeOn-GPT wine ontology exhibits approximately half the asserted classes found in gold standard wine ontology, suggesting a narrower scope of conceptual richness.
- NeOn-GPT wine ontology has approximately twice the object properties compared to gold standard wine ontology, suggesting better identification of salient features within the wine domain.
- NeOn-GPT wine ontology has twelve times the number of data properties compared to gold standard wine ontology, pointing to greater depth in class membership assignment.
- Although NeOn-GPT wine ontology has higher counts of object and data properties than the gold standard wine ontology, it features significantly fewer axioms. Reflecting that LLMs struggle to establish subject-relation-object triples. This can be attributed to LLMs generative capabilities that rely on statistical correlations and vector search methods, not on deductive reasoning or formal logic.
- NeOn-GPT wine ontology contains only one-sixth the number of individuals present in gold standard wine ontology, reflecting its stronger emphasis on identification rather than instance enumeration.

Results from hierarchical comparison illustrated in Fig. 10 and Table 1, can be summarised as:

- Class distribution: NeOn-GPT wine ontology differs notably from gold standard wine ontology, featuring a total of 39 classes, 36 classes without and 3 with sub-classes, compared to gold standard wine ontology with a total of 77 classes, 67 without and 10 with sub-classes.
- Class depth: NeOn-GPT wine ontology hierarchy has levels L0-L2 (`Thing -> Wine -> Red Wine`) versus gold standard wine ontology's L0-L3 more granular hierarchy (`Thing -> Potable Liquid -> Bordeaux -> Sauterness`).

6.2. Pre-inference and Post-inference Steps Assessment

Highlights the changes introduced by reasoning in both ontologies.

Pre-inference steps, NeOn-GPT's wine ontology allows instances of `"WhiteWine"` and `"RedWine"` to inherit from the `"Wine"` class due to their subsumption relationship. Conversely, the gold standard wine ontology uses class equivalence with color property restrictions, preventing `"WhiteWine"` and `"RedWine"` instances from inheriting `"Wine"` class memberships.

Post-inference steps, the gold standard wine ontology expands `"Wine"` instances, after including entities: `"WhiteWine"`, `"RedWine"`, and `"RoseWine"`. NeOn-GPT wine ontology only adds `"BurgundyRegion"` to `"Wine"`, showing limited classification properties by classifying based on basic concepts like geographical regions. Using the example of `"CabernetSauvignon"`, the gold standard wine ontology infers it as `"DryRedWine"` through class equivalence and restrictions, in contrast, NeOn-GPT wine ontology accurately, yet simplistically, infers it just as a `"GrapeVariety"` (see Fig. 11), reveals the limited depth of concept definition in the NeOn-GPT wine ontology for wine classification.

7. Conclusion and Future Work

We present a novel approach to ontology learning using LLMs and the NeOn Methodology Framework. Results from our experiments are presented in sections 5 and 6. Our results show that effective prompt engineering techniques alongside established ontology development methodologies can significantly influence LLMs' output to generate more consistent ontologies. However, experiments conducted in this study reveal that the inherent statistical representation of knowledge within LLMs cannot be directly converted to a formal representation for a domain-specific ontology creation with the same expressivity as the gold standard ontology for that domain. These limitations appear to stem from - class expressions and property restrictions. Class expressions in LLM-generated ontology are limited to subsumption type and lack conjunction, and disjunction expression types. This limits their ability to create complex classes from simple class definitions during the inference step. Similarly, property restriction types are limited to `"HasValue"` restrictions, the absence of cardinality, existential, and universal restrictions limit the scope of class membership inferences during the reasoning step. Our experiments demonstrate LLMs can be integrated into semi-automatic pipelines for generating base ontologies for enhancement through human-assisted knowledge. Recent studies with LLMs suggest they will eventually support Knowledge Engineering [35,21].

Future research work could explore augmenting our NeOn-GPT workflow with KG link prediction techniques to improve the expressivity and chain of reasoning for improving class expressions and property restrictions. Another path involves closely combining LLMs with workflows to create a system, acting as a copilot to support and guide users to build ontologies more effectively. Another approach, using pre-trained LLMs designed for code generation (e.g., Codex, Code LlamA) or fine-tuning pre-trained LLMs to produce valid and consistent ontologies could also enhance outcomes.

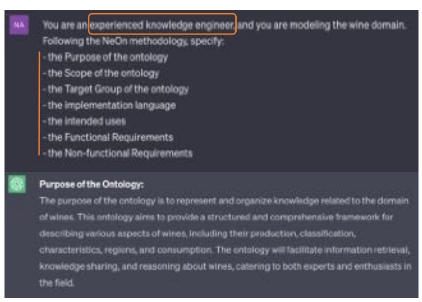
References

1. Auer, S.: The rapidowl methodology–towards agile knowledge engineering. In: 15th IEEE International Workshops on Enabling Technologies: Infrastructure for Collaborative Enterprises (WETICE'06). pp. 352–357. IEEE (2006)
2. Aussenac-Gilles, N., Biebow, B., Szulman, S.: Revisiting ontology design: a method based on corpus analysis. In: International Conference on Knowledge Engineering and Knowledge Management. pp. 172–188. Springer (2000)
3. Beckett, D., Berners-Lee, T., Prud'hommeaux, E., Carothers, G.: Rdf 1.1 turtle. World Wide Web Consortium pp. 18–31 (2014)
4. Bikeyev, A.: Synthetic ontologies: A hypothesis. Available at SSRN 4373537 (2023)
5. Chowdhery, A., Narang, S., Devlin, J., Bosma, M., Mishra, G., Roberts, A., Barham, P., Chung, H.W., Sutton, C., Gehrmann, S., et al.: Palm: Scaling language modeling with pathways. Journal of Machine Learning Research **24**(240), 1–113 (2023)
6. Cimiano, P., Völker, J.: Text2onto: A framework for ontology learning and data-driven change discovery. In: International conference on application of natural language to information systems. pp. 227–238. Springer (2005)
7. Fill, H.G., Fettke, P., Köpke, J.: Conceptual modeling and large language models: impressions from first experiments with chatgpt. Enterprise Modelling and Information Systems Architectures (EMISAJ) **18**, 1–15 (2023)
8. Gao, P., Han, J., Zhang, R., Lin, Z., Geng, S., Zhou, A., Zhang, W., Lu, P., He, C., Yue, X., et al.: Llama-adapter v2: Parameter-efficient visual instruction model. arXiv preprint arXiv:2304.15010 (2023)
9. Glimm, B., Horrocks, I., Motik, B., Stoilos, G., Wang, Z.: Hermit: an owl 2 reasoner. Journal of automated reasoning **53**, 245–269 (2014)
10. Haase, P., Herzig, D.M., Kozlov, A., Nikolov, A., Trame, J.: metaphactory: A platform for knowledge graph management. Semantic Web **10**(6), 1109–1125 (2019)
11. Haase, P., Lewen, H., Studer, R., Tran, D.T., Erdmann, M., d'Aquin, M., Motta, E.: The neon ontology engineering toolkit. WWW (2008)
12. Hearst, M.: Automated discovery of wordnet relations." wordnet an electronic lexical database (1998)
13. Ji, Z., Yu, T., Xu, Y., Lee, N., Ishii, E., Fung, P.: Towards mitigating LLM hallucination via self reflection. In: Bouamor, H., Pino, J., Bali, K. (eds.) Findings of the Association for Computational Linguistics: EMNLP 2023, Singapore, December 6-10, 2023. pp. 1827–1843. Association for Computational Linguistics (2023), `https://aclanthology.org/2023.findings-emnlp.123`
14. Joshi, I., Budhiraja, R., Akolekar, H.D., Challa, J.S., Kumar, D.: "it's not like jarvis, but it's pretty close!" - examining chatgpt's usage among undergraduate students in computer science. CoRR **abs/2311.09651** (2023). https://doi.org/10.48550/ARXIV.2311.09651, `https://doi.org/10.48550/arXiv.2311.09651`
15. Kojima, T., Gu, S.S., Reid, M., Matsuo, Y., Iwasawa, Y.: Large language models are zero-shot reasoners. Advances in neural information processing systems **35**, 22199–22213 (2022)
16. Krech, D., Grimnes, G.A., Higgins, G., Hees, J., Aucamp, I., Lindström, N., Arndt, N., Sommer, A., Chuc, E., Herman, I., Nelson, A., McCusker, J., Gillespie, T., Kluyver, T., Ludwig, F., Champin, P.A., Watts, M., Holzer, U., Summers, E., Morriss, W., Winston, D., Perttula, D., Kovacevic, F., Chateauneu, R., Solbrig, H., Cogrel, B., Stuart, V.: RDFLib (Aug 2023). https://doi.org/10.5281/zenodo.6845245, `https://github.com/RDFLib/rdflib`

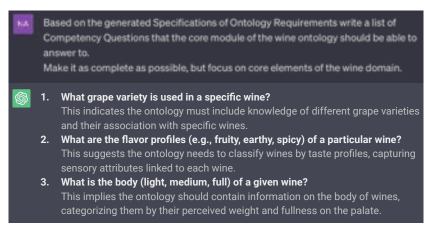
17. Maedche, A., Staab, S.: Ontology learning for the semantic web. IEEE Intelligent systems **16**(2), 72–79 (2001)
18. Marcus, G.: Sora's surreal physics. https://garymarcus.substack.com/p/soras-surreal-physics (2024), accessed: 2024-02-27
19. McGuinness, D.L., Van Harmelen, F., et al.: Owl web ontology language overview. W3C recommendation **10**(10), 2004 (2004)
20. Musen, M.A.: The protégé project: a look back and a look forward. AI Matters **1**(4), 4–12 (2015). https://doi.org/10.1145/2757001.2757003, https://doi.org/10.1145/2757001.2757003
21. Neuhaus, F.: Ontologies in the era of large language models - a perspective. Appl. Ontology **18**(4), 399–407 (2023). https://doi.org/10.3233/AO-230072, https://doi.org/10.3233/AO-230072
22. Noy, N.F., McGuinness, D.L., et al.: Ontology development 101: A guide to creating your first ontology (2001)
23. Pan, S., Luo, L., Wang, Y., Chen, C., Wang, J., Wu, X.: Unifying large language models and knowledge graphs: A roadmap. CoRR **abs/2306.08302** (2023). https://doi.org/10.48550/ARXIV.2306.08302, https://doi.org/10.48550/arXiv.2306.08302
24. Peroni, S.: 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 13. pp. 55–69. Springer (2017)
25. Petroni, F., Rocktäschel, T., Riedel, S., Lewis, P.S.H., Bakhtin, A., Wu, Y., Miller, A.H.: Language models as knowledge bases? In: Inui, K., Jiang, J., Ng, V., Wan, X. (eds.) Proceedings of the 2019 Conference on Empirical Methods in Natural Language Processing and the 9th International Joint Conference on Natural Language Processing, EMNLP-IJCNLP 2019, Hong Kong, China, November 3-7, 2019. pp. 2463–2473. Association for Computational Linguistics (2019). https://doi.org/10.18653/V1/D19-1250, https://doi.org/10.18653/v1/D19-1250
26. Poveda-Villalón, M., Gómez-Pérez, A., Suárez-Figueroa, M.C.: Oops!(ontology pitfall scanner!): An on-line tool for ontology evaluation. International Journal on Semantic Web and Information Systems (IJSWIS) **10**(2), 7–34 (2014)
27. Presutti, V., Daga, E., Gangemi, A., Blomqvist, E.: extreme design with content ontology design patterns. In: Proc. Workshop on Ontology Patterns. pp. 83–97 (2009)
28. Quelle, D., Bovet, A.: The perils & promises of fact-checking with large language models. CoRR **abs/2310.13549** (2023). https://doi.org/10.48550/ARXIV.2310.13549, https://doi.org/10.48550/arXiv.2310.13549
29. Shanahan, M., McDonell, K., Reynolds, L.: Role play with large language models. Nature **623**(7987), 493–498 (2023)
30. Smith, M., Zorpette, G., Choi, C.Q., Boyd, J.: Generative ai slims down for a portable world:¿ consumer tech aims llms everywhere-with laptops as the beachhead. IEEE Spectrum **61**(2), 5–13 (2024)
31. Spoladore, D., Pessot, E., Trombetta, A.: A novel agile ontology engineering methodology for supporting organizations in collaborative ontology development. Computers in Industry **151**, 103979 (2023)
32. Suárez-Figueroa, M.C., Gómez-Pérez, A., Fernandez-Lopez, M.: The neon methodology framework: A scenario-based methodology for ontology development. Applied ontology **10**(2), 107–145 (2015)

33. Tang, J., Yang, Y., Wei, W., Shi, L., Su, L., Cheng, S., Yin, D., Huang, C.: Graphgpt: Graph instruction tuning for large language models. arXiv preprint arXiv:2310.13023 (2023)
34. Tello, A.L., Gómez-Pérez, A.: ONTOMETRIC: A method to choose the appropriate ontology. J. Database Manag. **15**(2), 1–18 (2004). https://doi.org/10.4018/JDM.2004040101, `https://doi.org/10.4018/jdm.2004040101`
35. Wei, J., Tay, Y., Bommasani, R., Raffel, C., Zoph, B., Borgeaud, S., Yogatama, D., Bosma, M., Zhou, D., Metzler, D., et al.: Emergent abilities of large language models. arXiv preprint arXiv:2206.07682 (2022)
36. Wei, J., Wang, X., Schuurmans, D., Bosma, M., Xia, F., Chi, E., Le, Q.V., Zhou, D., et al.: Chain-of-thought prompting elicits reasoning in large language models. Advances in Neural Information Processing Systems **35**, 24824–24837 (2022)
37. Ye, J., Chen, X., Xu, N., Zu, C., Shao, Z., Liu, S., Cui, Y., Zhou, Z., Gong, C., Shen, Y., et al.: A comprehensive capability analysis of gpt-3 and gpt-3.5 series models. arXiv preprint arXiv:2303.10420 (2023)
38. Zhou, Y., Muresanu, A.I., Han, Z., Paster, K., Pitis, S., Chan, H., Ba, J.: Large language models are human-level prompt engineers. arXiv preprint arXiv:2211.01910 (2022)

A. Appendix A: Figures

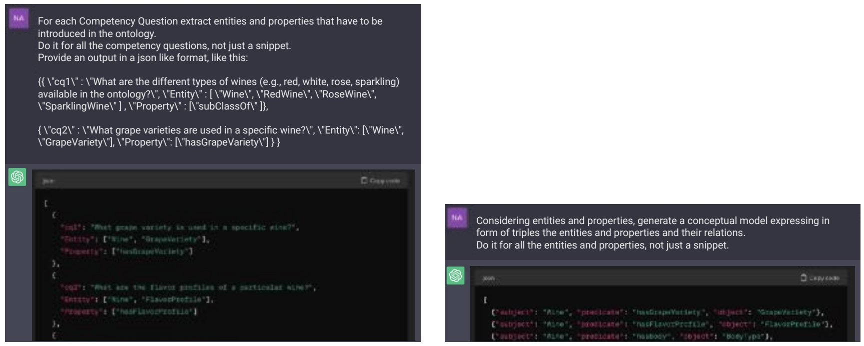


(a) Using Chain-of-Thought prompting to prompt ChatGPT to specify the wine ontology requirements following the NeOn Methodology Framework.



(b) Prompting ChatGPT to generate competency questions for the wine ontology.

Fig. 2: Step - 1 Ontology draft generation: specification of ontology requirements, prompting ChatGPT to specify the wine ontology requirements following the NeOn methodology framework.



(a) Using Few-shot prompt- ing to prompt ChatGPT to extract entities and properties.

(b) Prompting ChatGPT to generate the corresponding triples to the extracted entities and properties.

Fig. 3: Step - 1 Ontology draft generation: prompting ChatGPT to create a conceptual model of the wine ontology based on the generated competency questions in Fig. 2b.

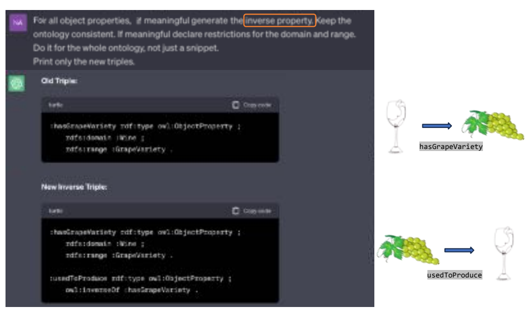


Fig. 4: Step - 1 Ontology draft generation: illustration of prompting ChatGPT to extend the generated wine ontology with inverse object properties: "`usedToProduce`" inverse to the existing "`hasGrapeVariety`" property.

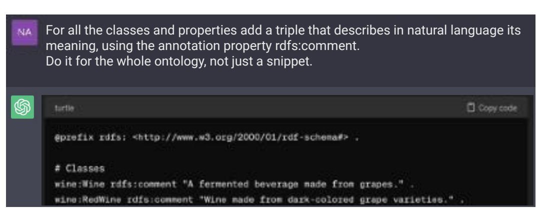


Fig. 5: Step - 1 Ontology draft generation: prompting ChatGPT to generate natural language descriptions for entities and properties in the wine ontology.

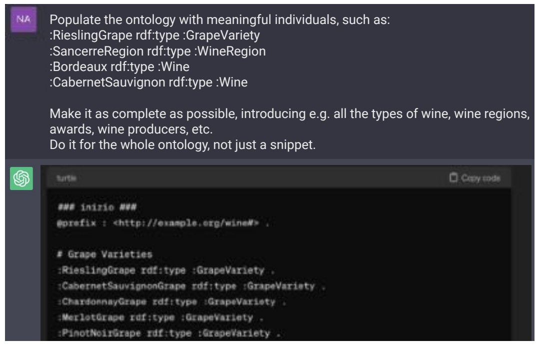


Fig. 6: Step - 1 Ontology draft generation: using Few-shot prompting to prompt ChatGPT to populate the wine ontology with real world instances.

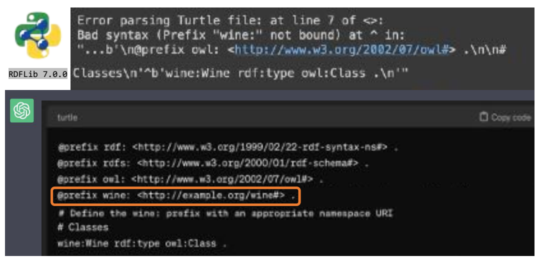


Fig. 7: Step - 2 Syntax validation: example of using errors produced by RDFLib to prompt ChatGPT to fix syntax errors, declaring undefined prefix `"Wine"`.

Fig. 8: Step - 3 Consistency check: illustration of an inconsistency in the wine ontology detected by HermiT Reasoner, incorrect assignment of `"Yellow"` as a wine color. ChatGPT identified `"White"` as the valid color using the HermiT reasoner error as a prompt, aligning with the domain range: `(Red, White, Rosé)`.

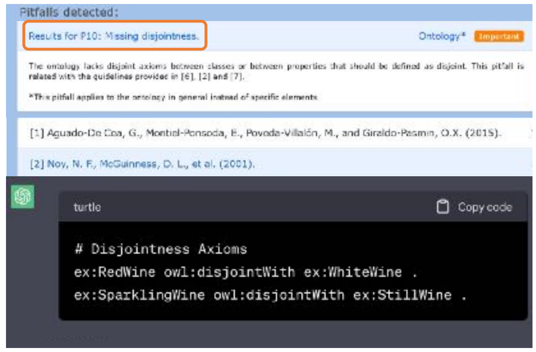
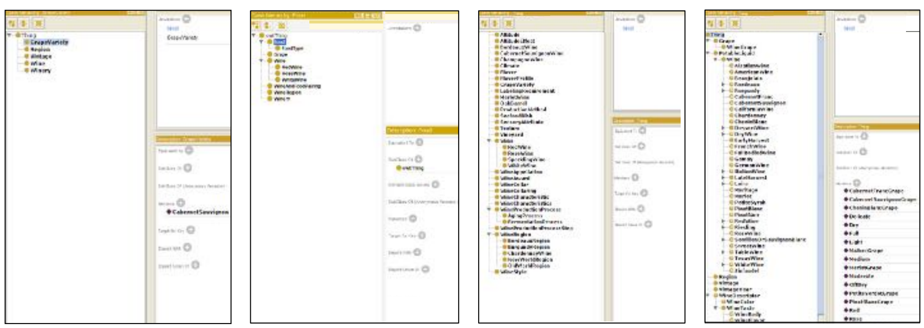
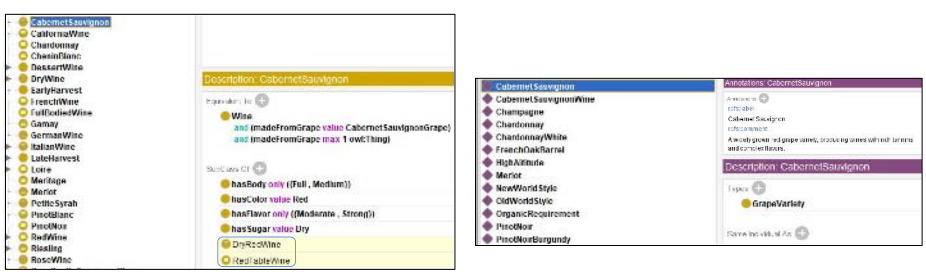


Fig. 9: Step - 4 Pitfall resolution: illustration of a pitfall detected by OOPS API where the wine ontology lacked disjointness. Prompting ChatGPT using the pitfall description from OOPS API as a prompt, ChatGPT introduced the necessary disjointness axioms such as "`RedWine`" is disjoint with "`WhiteWine`".



(a) Zero-shot Prompting (b) Ontology Development Guide Prompts (c) NeOn-GPT (d) Gold Standard (Benchmark)

Fig. 10: Comparative overview of the wine ontology structures.



(a) Gold Standard Inferred Instance (b) NeOn-GPT Inferred Instance

Fig. 11: Ontological inferences of "`CabernetSauvignon`".

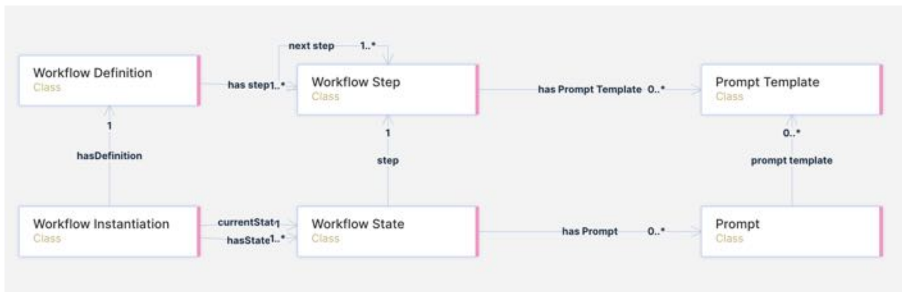
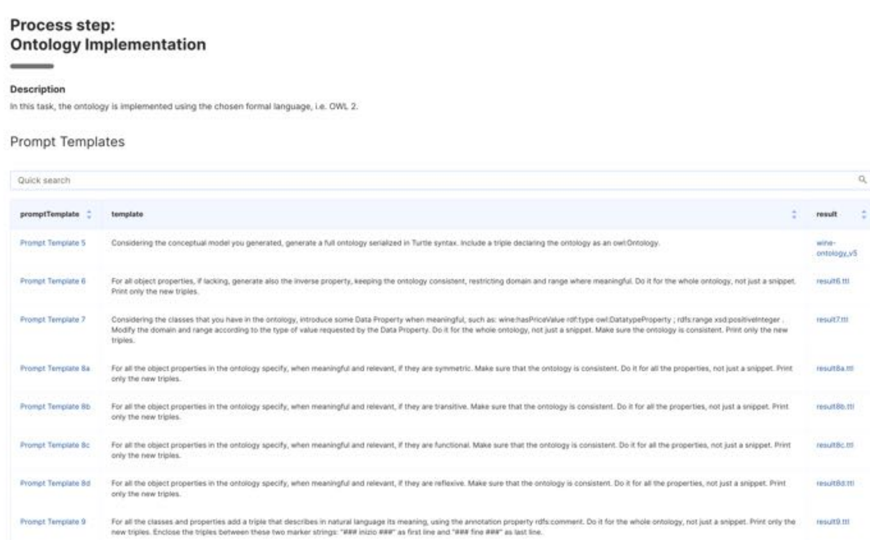


Fig. 12: metaphactory built-in Workflow Ontology. *Workflow Definition* and *Workflow Instantiation* are distinguished. A workflow is defined by a sequence of *Workflow steps*, where each step has one or more *Prompt Templates*, applied in a particular order. A workflow instance is a sequence of *Workflow states*, where each state has one or more *prompts*.



Process step:
Ontology Implementation

Description

In this task, the ontology is implemented using the chosen formal language, i.e. OWL 2.

Prompt Templates

Quick search

promptTemplate	template	result
Prompt Template 5	Considering the conceptual model you generated, generate a full ontology serialized in Turtle syntax. Include a triple declaring the ontology as an owl:Ontology.	wine-ontology_v5
Prompt Template 6	For all object properties, if lacking, generate also the inverse property, keeping the ontology consistent, restricting domain and range where meaningful. Do it for the whole ontology, not just a snippet. Print only the new triples.	result6.ttl
Prompt Template 7	Considering the classes that you have in the ontology, introduce some Data Property when meaningful, such as: wine:hasPriceValue rdf:type owl:DatatypeProperty ; rdfs:range xsd:positiveInteger . Modify the domain and range according to the type of value requested by the Data Property. Do it for the whole ontology, not just a snippet. Make sure the ontology is consistent. Print only the new triples.	result7.ttl
Prompt Template 8a	For all the object properties in the ontology specify, when meaningful and relevant, if they are symmetric. Make sure that the ontology is consistent. Do it for all the properties, not just a snippet. Print only the new triples.	result8a.ttl
Prompt Template 8b	For all the object properties in the ontology specify, when meaningful and relevant, if they are transitive. Make sure that the ontology is consistent. Do it for all the properties, not just a snippet. Print only the new triples.	result8b.ttl
Prompt Template 8c	For all the object properties in the ontology specify, when meaningful and relevant, if they are functional. Make sure that the ontology is consistent. Do it for all the properties, not just a snippet. Print only the new triples.	result8c.ttl
Prompt Template 8d	For all the object properties in the ontology specify, when meaningful and relevant, if they are reflexive. Make sure that the ontology is consistent. Do it for all the properties, not just a snippet. Print only the new triples.	result8d.ttl
Prompt Template 9	For all the classes and properties add a triple that describes in natural language its meaning, using the annotation property rdfs:comment. Do it for the whole ontology, not just a snippet. Print only the new triples. Enclose the triples between these two marker strings: "### inizio ###" as first line and "### fine ###" as last line.	result9.ttl

Fig. 13: NeOn-GPT metaphactory Integration: Ontology Implementation Step.