

OntoLearner: A Modular Python Library for Ontology Learning with Large Language Models

Hamed Babaei Giglou

`hamed.babaei@tib.eu`

TIB - Leibniz Information Centre for Science and Technology <https://orcid.org/0000-0003-3758-1454>

Jennifer D'Souza

TIB – Leibniz Information Centre for Science and Technology

Andrei Aioanei

TIB - Leibniz Information Centre for Science and Technology

Nandana Mihindukulasooriya

IBM Research

Sören Auer

TIB – Leibniz Information Centre for Science and Technology

Article

Keywords:

Posted Date: June 25th, 2026

DOI: <https://doi.org/10.21203/rs.3.rs-9539494/v1>

License: © ⓘ This work is licensed under a Creative Commons Attribution 4.0 International License.

[Read Full License](#)

Additional Declarations: There is **NO** Competing Interest.

OntoLearner: A Modular Python Library for Ontology Learning with Large Language Models

Hamed Babaei Giglou^{1,*} Jennifer D’Souza^{1,*} Andrei Aioanei¹

Nandana Mihindukulasooriya³ Sören Auer^{1,2}

¹TIB - Leibniz Information Centre for Science and Technology, Hannover, Germany

²L3S Research Center, Leibniz University of Hannover, Hannover, Germany

³IBM Research, New York, USA

¹{hamed.babaei,jennifer.dsouza,andrei.aioanei,auer}@tib.eu

³nandana@ibm.com

Abstract

Ontology learning (OL) aims to automatically construct structured knowledge models from text, yet progress remains fragmented across methods, domains, and evaluation practices. Despite decades of research, OL lacks a shared infrastructure for systematic evaluation and ontology access. This absence has hindered progress and fragmented research, leaving the central challenges of OL largely unaddressed. We introduce OntoLearner, a modular, cross-domain, and first-of-its-kind framework that unifies ontology access, large language model (LLM)-driven learning pipelines, and standardized benchmarking. OntoLearner releases 180 machine-readable ontologies spanning 22 domains and provides pipeline-ready datasets with train/dev/test splits for three core OL tasks: term typing, taxonomy discovery, and non-taxonomic relation extraction. Using this infrastructure, we conduct a large-scale empirical study of OL, evaluating 22 retrieval models and 12 LLMs across domains and tasks. The results converge on a finding that reframes the central challenge of OL: failure modes scale with ontological complexity rather than model size or architectural sophistication. The primary bottleneck is not model capability, but a structural mismatch between how models encode knowledge and how ontologies organize it. These findings establish that effective OL is reachable through the cross-domain, multi-task benchmarking enabled by OntoLearner. OntoLearner is open-source (MIT license) at <https://github.com/sciknoworg/OntoLearner/>.

1 Introduction

Ontology learning (OL) [67] sits at the intersection of symbolic and statistical artificial intelligence (AI), addressing a long-standing challenge in the field: how to combine structured, interpretable knowledge with the flexibility and scalability of data-driven models. Ontologies provide explicit, formal representations of concepts and their relationships, forming a foundation for reasoning, explainability, and interoperability in complex AI systems [107]. In emerging neuro-symbolic paradigms [50, 11], such structured knowledge plays a critical role in grounding learning, constraining generation, and enabling more robust generalization beyond surface-level correlations. However,

*These authors contributed equally to the work.

Table 1: Comparative positioning of OntoLearner.

(a) Feature comparison of OntoLearner against major ontology infrastructure categories: OD (Ontology Development and Repository Platforms), OLS (Ontology Access and Lookup Services), and VI (Vocabulary and Terminology Integration Services). $\sim$ indicates planned functionality.

Feature	OD	OLS	VI	OntoLearner
Cross-Domain Coverage	✗	✓	✓	✓
Ontology Hosting and Versioning	✓	✗	✗	✓
Search and Browse Functionality	✓	✓	✓	✓
Programmatic Access (API)	✓	✓	✗	✓
Vocabulary Alignment and Mapping	✓	✗	✓	$\sim$
LLM Integration	✗	✗	✗	✓
Ontology Editing/Authoring Tools	✓	✗	✗	✗
Modular and Reusable Components	✗	✗	✗	✓
Support for Ontology Enrichment	✓	✗	✗	✓
FAIR Principles Alignment	✓	✓	✓	✓

(b) Comparison of OntoLearner with recent AI/LLM-based ontology learning systems. “Bench.” indicates provision of standardized benchmarks or dataset splits.

System	Main Tasks	LLM Integration	Domain Scope	Bench.
NeOn-GPT [32]	Requirements, class hierarchy, OWL axioms	GPT-3.5 prompting	Specific (Wine ontology)	✗
OntoGPT [17]	Entity/relation extraction	Schema-driven zero-shot prompting	Biomedical, recipes	✗
OntoChat [123]	Requirement engineering (e.g., CQs)	Interactive prompting	General (e.g., Music)	✗
OLLM [65]	Taxonomy induction	Fine-tuned LLM	Cross-domain (Wikipedia, ArXiv)	✗
Ontolearn [23]	OWL class expression learning	LLM-based verbalization (support module)	Large KGs (symbolic focus)	✗
OntoLearner (ours)	Term typing, taxonomy, relations	Modular prompting, hybrid workflows	22 domains (180 ontologies)	✓

29 constructing and maintaining high-quality ontologies remains a bottleneck, limiting their integration
30 into modern AI pipelines.

31 An ontology engineer may need to determine whether “cell death” should be classified as a “bio-
32 logical process”, a “pathological state”, or a “causal event”. Although seemingly narrow, such a
33 decision propagates across every system that relies on the ontology: databases that query it, mod-
34 els that reason over it, and scientists who reuse it. Across thousands of similar decisions spanning
35 hundreds of domains—expanding faster than expert teams can manage—this situation illustrates
36 the central challenge of OL: transforming heterogeneous natural language into structured, machine-
37 interpretable conceptual models that enable reasoning, interoperability, and scientific discovery.

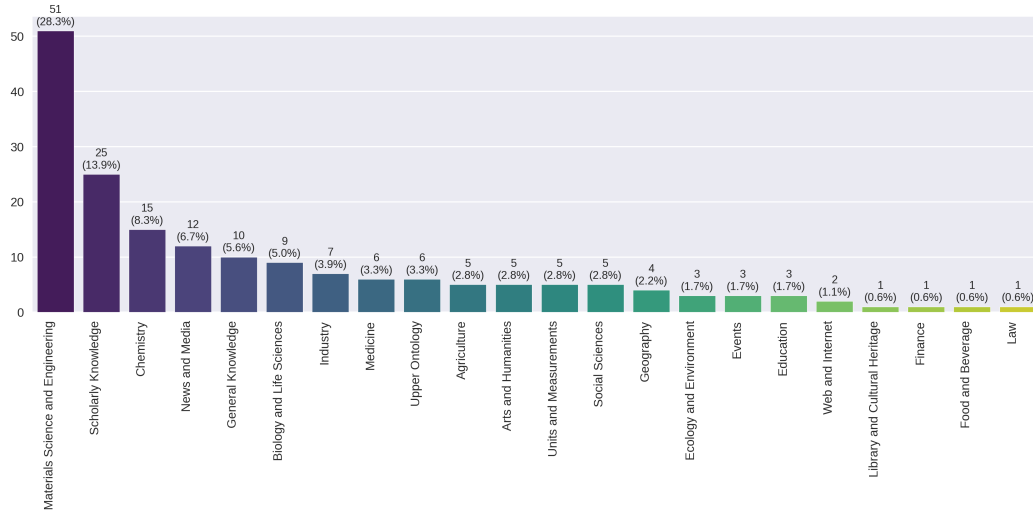
38 Over the past three decades, numerous approaches have attempted to automate this challenge (see
39 Appendix A for detailed related works). The first wave brought lexico-syntactic patterns [46]
40 and modular pipelines: Text2Onto [19] combined probabilistic modeling with linguistic heuris-
41 tics, OntoLearn [74, 114] added syntactic parsing and graph-based pruning, and OntoGen [34] in-
42 troduced interactive user-in-the-loop refinement. These were principled systems, but often brittle.
43 They struggled under terminological variation, struggled to align with established vocabularies, and
44 produced ontologies too inconsistent to reuse reliably across domains. The second wave scaled
45 up: NELL [16] continuously populated seed ontologies via semi-supervised learning, Klink-2 [86]
46 mined co-occurrences and citations to construct a 14,000-topic knowledge structure, and industry
47 deployments, such as Microsoft Probase [119], Diffbot [27], and PoolParty [103], leveraged web-
48 scale extraction and graph-based methods. However, increased scale did not compensate for the
49 absence of logical rigor, often resulting in large but weakly structured knowledge collections. In
50 parallel, the community built ontology support infrastructure, including BioPortal [80, 117, 12], the
51 Open Biomedical Ontologies (OBO) Foundry [105], Ontology Lookup Service (OLS) [56, 37], the
52 TIB Terminology Service [61, 111, 25], Linked Open Vocabularies (LOV) [113], Unified Medical
53 Language System (UMLS) [13], and FAIRsharing [97]—essential repositories for hosting, discover-
54 ing, reusing, and aligning ontologies across domains. Yet none was designed to evaluate or auto-
55 mate the learning process itself [58]. Then, with the emergence of large language models (LLMs), a
56 new wave of ontology learning approaches has emerged: NeOn-GPT [32] used structured prompts
57 to elicit concept hierarchies, OntoGPT [17] applied schema-based extraction via recursive zero-
58 shot prompting, OntoChat [123] introduced conversational interfaces for early modeling tasks, and
59 OLLM [65] generated entire taxonomies end-to-end from domain corpora. Hybrid pipelines that
60 combine embeddings with few-shot prompting outperformed single-method baselines [38, 39], and

human-LLM collaboration reduced extraction effort by over 50% [60]. Yet logical inconsistencies persisted [32], outputs remained domain-narrow [17], semantic drift proved difficult to control [60]. Our early benchmarking efforts in LLMs4OL initiative [8, 38, 39] and efforts to benchmark LLMs systematically on OL tasks confirmed this directly: foundation models can assist with OL tasks (i.e., term typing, taxonomy induction, and relation extraction), but no shared framework has been established to evaluate, compare, or build upon these capabilities across domains. Taken together, these limitations point to a broader gap: *the absence of a unified infrastructure capable of grounding, integrating, and systematically evaluating automated OL across methods and domains*.

The Table 1a summarizes and compares the strengths of each infrastructure category, whereas the Table 1b contrasts AI/LLM-based learning systems. The pattern reveals a deeper problem than any single method could solve. Automation tools and ontology infrastructure evolved in isolation—one side building learners, the other building repositories, neither building the bridge between them. LLMs added generative power but no formal grounding. And critically, across all three waves, there was never a unified framework for measuring progress: no shared benchmarks, no standardized splits, no cross-domain evaluation. Without standardized evaluation, progress across iterations remained difficult to accumulate.

We introduce **OntoLearner**, a modular, cross-domain OL framework that addresses this gap by unifying, for the first time, ontology access, AI/LLM-driven learning, and systematic benchmarking into a single coherent infrastructure. OntoLearner builds upon the ontology infrastructure ecosystem, particularly BioPortal, OLS, OBO Foundry, LOV, and FAIRsharing, and, as a first-of-its-kind, releases 180 ontologies across 22 domains as machine-readable, pipeline-ready, and LLM-compatible datasets hosted on HuggingFace, directly bridging the gap between ontology repositories and modern AI pipelines. Unlike prior ontology hubs that focus on browsing and discovery, OntoLearner enables standardized loading, versioning, and streaming across Python workflows, making it well-suited for seamless integration with pretrained models, tokenizers, and evaluation scripts, supporting reproducible and scalable OL. Rather than replacing human expertise, OntoLearner positions LLMs as assistive components grounded in curated vocabularies, combining the rigor of symbolic knowledge engineering with the generalization power of foundation models, and providing the infrastructure to measure how well that combination actually works. Concretely, OntoLearner delivers four contributions: (1) a unified interface for transforming ontologies from diverse repositories into machine-readable formats suitable for LLM applications; (2) a modular library supporting core OL tasks (including term typing, taxonomy discovery, and non-taxonomic relation extraction) with standardized train/dev/test splits and evaluation metrics enabling rigorous, reproducible comparison across classical, neural, and hybrid methods; (3) AI/LLM integration as assistive components aligned with curated vocabularies, supporting concept suggestion, term typing, hierarchy completion, and definition generation grounded in existing knowledge models; and (4) an extensible, domain-agnostic platform aligned with FAIR principles, sustained through HuggingFace², GitHub³, and complete documentation⁴ to ensure long-term accessibility and community-driven development.

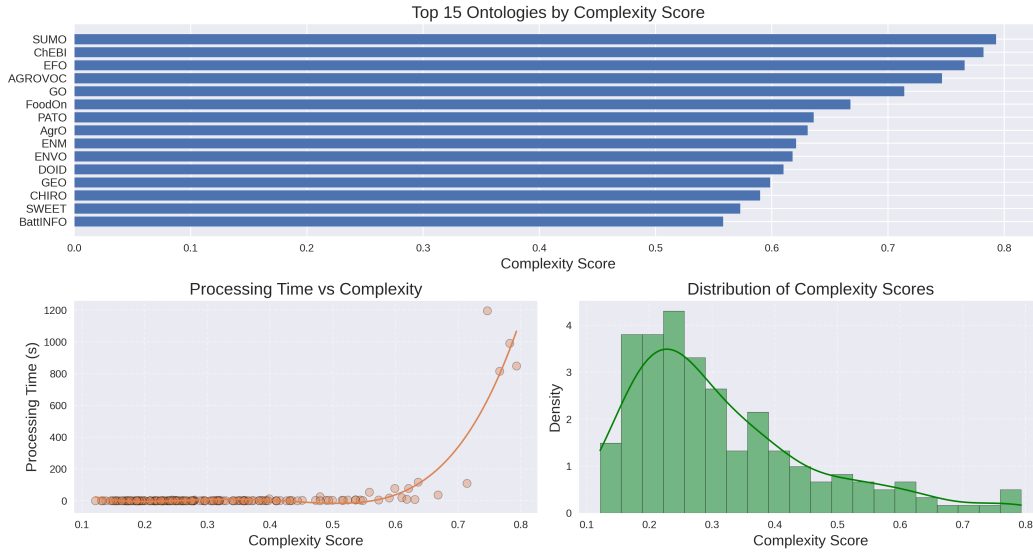
OntoLearner is a standalone OL library rather than a wrapper around HuggingFace, that introduces high-level abstraction layers bridging symbolic and neural methods, with a long-term vision of becoming a standard library for OL, serving semantic web and machine learning researchers and practitioners. Beyond providing infrastructure, OntoLearner enables systematic empirical study of OL across domains. Using this framework, we show evidence of how classical and modern embedding models, as well as LLM-based approaches, behave differently across OL tasks, where semantic drift and logical inconsistencies emerge, and how grounding in curated ontologies affects these outcomes. These findings transform OntoLearner from a tooling contribution into a basis for reproducible scientific evaluation of OL methods, advancing the field from exploratory prototyping toward trustworthy, AI-assisted ontology engineering.



(a) Distribution of 180 benchmark ontologies across 22 domains. The number of ontologies is mentioned above the bars with their percentage in comparison to the total ontologies in the collection.

(b) Summary statistics of key ontology metrics in OntoLearner, including mean, standard deviation (STD), quartiles (25%, 50%, 75%), and maximum values.

Metric	Mean	STD	25%	50%	75%	Max
<i>Total Nodes</i>	43,455	260,522	227	659	4,245	2,433,610
<i>Class No.</i>	3,401	18,672	30	99	610	220,816
<i>Properties No.</i>	93	266	10	35	93	3,029
<i>Individuals No.</i>	7,654	92,233	0	0	15	1,234,769
<i>Averaged Depth</i>	2.27	3.62	0.57	1.37	2.71	38.24
<i>Averaged Breadth</i>	3,852	27,830	5	16	63	310,545
<i>Processing Time (s)</i>	25.12	144.33	0.03	0.10	0.95	1196.53



(c) Structural Analysis: The top plot shows the top-15 ontologies' complexity scores, where the bottom-left plot shows the processing time vs. complexity score behaviour, and the bottom-right plot shows the distribution of complexity scores of ontologies.

Figure 1: Overview of the OntoLearner benchmark collection.

2 Results

2.1 Overview of Available Ontologies

The ontologies included in OntoLearner span diverse domains such as materials science and engineering, biology, life sciences, agriculture, law, library science, and cultural heritage. As of the date of this article submission, the collection comprises 180 ontologies containing 612,223 classes across 22 domains, and is designed to grow continuously. Ontologies were systematically collected using four objectives: 1) domain representativeness across major scientific disciplines, 2) structural diversity, 3) accessibility and licensing compatibility, and 4) active maintenance status. Each ontology was processed with OntoLearner’s *Ontologizer* component (described in subsubsection 4.2.1) to ensure consistent metric computation and programmatic access. Figure 1a depicts the distribution of ontologies. The *Materials Science and Engineering*, with ontologies (28.3%), is the most represented domain, followed by Scholarly Knowledge with 25 ontologies (13.9%), while the 13 domains contain five or fewer ontologies each.

An ontology can be statistically presented via multiple metrics including graph metrics (i.e., nodes, edges), knowledge coverage metrics (i.e., classes, individuals, properties), hierarchical metrics (i.e., max, min, and mean depth), breadth metrics (i.e., max, min, and mean breadth), and dataset statistics (i.e., term types, taxonomic relations). All metrics are automatically updated as new ontologies are added and are publicly accessible via a dedicated benchmark metrics space⁵ and documentation website. A summary of these statistics is presented in Figure 1b, revealing substantial structural heterogeneity across the collection (high variability in STD). Most ontologies are small to medium in size, with the median substantially smaller than the mean, reflecting a right-skewed distribution. However, a small subset is extremely large, deep, and richly connected. This long-tail distribution reflects realistic ontology ecosystems, where lightweight vocabularies coexist with large, highly expressive knowledge models.

To further capture ontological variability, OntoLearner introduces an ontological complexity score, C_{score} (described in subsubsection 4.2.3), that summarizes structural characteristics into a single value. The structural analysis in Figure 1c highlights how this score differentiates ontologies across the collection. Highly expressive ontologies such as SUMO [78], ChEBI [68], EFO [69], AGROVOC [108], and GO [2, 20] appear among the most complex ($C_{score} \approx 0.72\text{--}0.82$), reflecting their large class counts, deep hierarchies, and extensive branching structures (see top plot in Figure 1c). The relationship between C_{score} and processing time (bottom-left plot) shows a clear nonlinear trend: ontologies with $C_{score} < 0.5$ require minimal processing time, whereas computational cost increases sharply beyond $C_{score} \approx 0.6$, with the most complex ontologies requiring up to 1,200 seconds to process. The distribution of C_{score} values across the collection (bottom-right plot) further confirms the long-tail pattern observed in the statistical summary, where most ontologies fall within moderate complexity ranges (0.15–0.35) and only a small subset are structurally and computationally demanding. Overall, this diversity in size, structure, and computational cost makes the OntoLearner collection particularly suitable for benchmarking OL methods, as it reflects the wide spectrum of ontology characteristics encountered in real-world knowledge engineering scenarios.

2.2 Retrievers

Effective OL depends on retrieving relevant ontological knowledge before any generation or classification. We evaluate 22 retrieval models spanning lexical baselines, classical embeddings, sentence transformers, dense retrievers, and domain-specialized models across the three OL tasks (learning tasks are detailed at subsubsection 4.2.1), using *Recall@15* as the primary metric. This broad comparison reveals where retrieval alone suffices, where it fails, and where LLM-Augmented retrieval helps (experimental setups are provided at subsection 4.4). Results are reported on 26 ontologies selected to represent the structural diversity of the OntoLearner collection (see subsubsection 4.4.2).

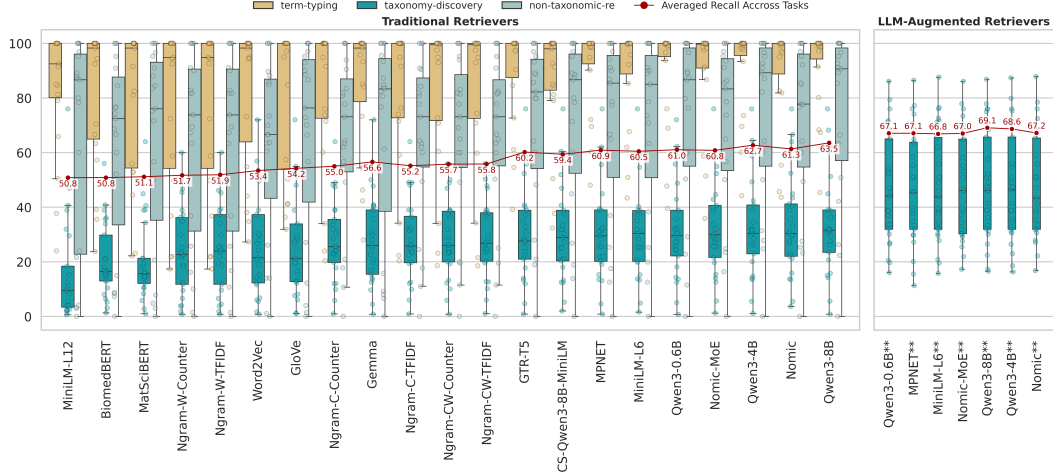
Task-Dependent Performance. Results in Figure 2a show clear task dependence. Term typing is generally easy, with most embedding models achieving high recall (median >95%), with Qwen3

²<https://huggingface.co/collections/SciKnowOrg/ontolearner-benchmarking-6823bcd051300c210b7ef68a>

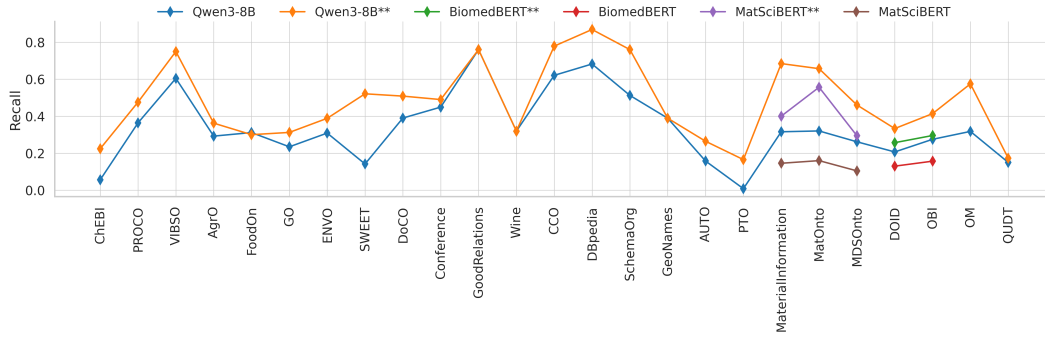
³<https://github.com/sciknoworg/OntoLearner/>

⁴<https://ontolearner.readthedocs.io/>

⁵<https://huggingface.co/spaces/SciKnowOrg/OntoLearner-Benchmark-Metrics>



(a) Recall performance of various retrieval models using top-k 15, by task. Boxplots summarize the distribution per model per task for our selected collection of 26 ontologies. The dots show individual recall per ontology. The models with the ** symbol refer to the results of the augmented retriever approach. Retrievers on the x-axis are ordered by average recall of the type taxonomy discovery task. The red line chart represents each retriever’s averaged recall across all three tasks. Augmented retrievers are only applied to the type taxonomy discovery task.



(b) Recall comparison across ontologies for the taxonomy discovery task using the traditional *Qwen3-8B* retriever and the LLM-augmented variant *Qwen3-8B***. Domain-specific retrievers (*BiomedBERT* and *MatSciBERT*) are also shown where applicable. Results are reported using a top-*k* value of 15. Ontologies are represented by their OntoLearner IDs on the x-axis; the corresponding full ontology names are listed in Table 5.

Figure 2: Retriever-based learners comparison.

(0.6B, 4B, 8B) and Nomic variants averaging 89–92%. Taxonomy discovery is much harder, with top models reaching only 9–37% recall, reflecting the difficulty of capturing hierarchical relations through embeddings alone [121, 7, 101]. Among classic retrievers, Qwen3-8B performs best for taxonomy discovery (averaged recall of $\approx 34\%$). Non-taxonomic RE shows the greatest variability (near 0 to higher than 90%), depending on domain and relation type, highlighting that some relations align well with embeddings (e.g., part-of, used-for) while others remain highly context-dependent. This gap confirms that retrieval alone cannot solve OL tasks.

Impact of Model Architecture and Scale. Lexical baselines (i.e, *Ngrams*) achieve moderate recall on term typing (averaged recall of $\approx 80\%$) and non-taxonomic relations ($\approx 64\%$) but perform poorly on taxonomy discovery ($\approx 27\%$), illustrating the limits of surface-form approaches. Traditional embeddings (GloVe, Word2Vec) show similar patterns ($\approx 81\%$, 26% , 63%), though they suffer from out-of-vocabulary. Sentence transformers improve substantially ($\approx 90\%$, 37% , 70%), with larger models like GTR-T5 and MPNET outperforming smaller MiniLM. Modern embeddings reach slightly higher recall ($\approx 91\%$, 40% , 71%), with Qwen3-8B providing the best balance. However, scaling across 0.6B–8B yields diminishing returns. Nomic and Nomic-MoE perform similarly, demonstrating that bigger (8B) or more complex models (Nomic-MoE) alone cannot fully solve

these tasks. Even though the simple embedding models work well for simple tasks, they hit a ceiling on taxonomy discovery.

Domain-Specific Retrievers. Domain-specialized models such as BiomedBERT [42] and MatSciBERT [43] perform poorly on general-domain ontologies, with an average recall of $\approx 51\%$. Although pretrained on specialized corpora and strong on domain-focused NLP tasks, their retrieval results in OL indicate limited preservation of structured semantics. The Figure 2b further shows that LLM-based enhancement yields uneven improvements. While augmentation produces measurable gains, particularly within its respective domain, cross-domain generalization remains weaker than that of general-purpose retrievers. These findings suggest domain-specific pretraining biases the embedding space toward specialized terminology and local semantic patterns, hindering alignment with heterogeneous ontology structures.

Cross-Task Performance Consistency. The red line in Figure 2a shows averaged recall across all tasks, highlighting model robustness. Qwen3-8B and Nomic are the most consistent, maintaining high recall across diverse tasks. In contrast, specialized models exhibit task-specific variability: character-level n-grams excel at term typing but perform poorly on relational tasks. The hybrid CS-Qwen3-8B-MiniLM performs well on term typing and non-taxonomic relations but offers a marginal gain on taxonomy discovery over Qwen3-8B, suggesting that architectural complexity does not compensate for the task’s difficulty. Nevertheless, no single architectural family consistently maintains strong performance across OL tasks. The gap between task-specific peaks and cross-task consistency reveals that single-method optimization systematically trades robustness for narrow performance gains.

Breaking the Retrieval Ceiling: LLM-Augmented Retrieval. Applying LLM-Augmented retrieval to taxonomy discovery substantially boosts recall across top retrievers, improving average performance by 14–16%. Notable gains include Nomic** (33.2% $\rightarrow$ 47.9%), Qwen3-4B** (32.9% $\rightarrow$ 47.9%), and Qwen3-8B** (33.8% $\rightarrow$ 47.7%). Mid-tier models benefit most: Nomic-MoE** (+15.6%) and MiniLM-L6** (+16%), narrowing the gap with top performers. Qwen3-8B** achieves the highest overall recall ($\approx 69.1\%$, +5.6% over base). By generating plausible parent candidates with an LLM before embedding-based ranking, this two-stage approach combines generative reasoning with discriminative retrieval, surpassing either component alone and exemplifying the hybrid infrastructure OntoLearner was designed to enable.

These results reveal a consistent pattern: *no single retrieval architecture, regardless of scale, specialization, or complexity, can handle all OL tasks*. Term typing aligns with embeddings; taxonomy discovery does not; domain pretraining limits generalization; and scaling offers diminishing returns. These observations indicate that effective OL requires principled combinations of retrieval and generation. LLM-Augmented retrieval demonstrates that such hybrid approaches are both feasible and superior, and OntoLearner provides the infrastructure to evaluate and compare such approaches systematically.

2.3 Reranking for the RAG Pipeline

Table 2 presents averaged F1-scores of the end-to-end RAG pipeline across domains, using *Qwen3-Embedding-8B* as a retriever and 12 LLMs as rerankers. Unlike our initial LLMs4OL work [8], which constrained the search space, candidates come from the full ontology class space, simulating realistic conditions where models must identify correct relations among thousands of plausible alternatives [38, 39]. This change shifts the difficulty landscape: non-taxonomic RE was hardest in our earlier work, but taxonomy discovery is now the dominant challenge, reflecting the complexity of hierarchical reasoning at scale.

Task Difficulty Reflects Real-World Complexity. A clear performance hierarchy emerges across tasks. Term typing achieves the highest F1-scores (40–83%; e.g., Qwen3-Next-80B reaches 82.7% in *Finance*), taxonomy discovery performs poorly (single digits in *Biology & Life Science*, *Agriculture*, and *Ecology & Environment*), and non-taxonomic RE falls in between, ranging from 0.2% (*Agriculture*) to 95.7% (*Units & Measurements*, Falcon-H1-1.5B). Variability reflects task characteristics: in high-frequency domains like *Units & Measurements*, LLMs memorize relations, making retrieval mostly redundant, whereas sparse domains like *Agriculture* benefit most from RAG, which bridges partial model knowledge and task requirements. Term typing is effectively a local classification problem, while taxonomy discovery requires inferring a global hierarchical structure. Neither

Table 2: Averaged F1-scores (%) of end-to-end RAG pipeline using Qwen3-Embedding-8B as a retriever across multiple domains and tasks. For each domain task, the F1-scores are averaged across all available ontologies. The number of ontologies per domain are: *Events* (1), *Finance* (1), *Food & Beverage* (1), *Agriculture* (2), *Geography* (1), *Materials Science & Engineering* (4), *Biology & Life Sciences* (1), *Ecology & Environment* (2), *General Knowledge* (3), *Industry* (2), *Medicine* (2), *Units & Measurements* (2), *Chemistry* (3), and *Education* (1). Best, second-best, and third-best results per task are highlighted using green (▲), blue (●), and red (◆), respectively. The best-performing LLM per task is indicated in bold, while the best-performing LLM per domain is underlined. The “I” refers to the instruction model, where the “TH” refers to the thinking variant.

	Qwen3-0.6B	Qwen3-8B	Qwen3-14B	Qwen3-4B-I	Qwen3-4B-TH	Qwen3-Next-80B-I	Gemma-3-1B-it	Gemma-3-4B-it	Gemma-3-12B-it	Gemma-3-27B-it	Falcon-H1-1.5B-Deep-I	Mistral-Small-3.2-24B-I
Term Typing Task												
<i>Events</i>	50.0	56.2	40.0	58.1	49.1	55.2	43.2	48.5	52.4	56.0	48.8	53.1
<i>Finance</i>	22.2	65.6	71.8	71.2	25.0	▲ 82.7	21.4	65.7	62.3	64.2	54.5	68.3
<i>Food & Beverage</i>	8.5	54.1	42.0	61.0	7.3	<u>61.3</u>	11.9	29.3	30.8	49.5	47.3	42.7
<i>Agriculture</i>	39.5	<u>69.5</u>	61.9	60.5	36.6	<u>67.6</u>	33.4	47.5	55.9	58.6	52.4	61.9
<i>Geography</i>	66.7	◆ 81.9	5.0	● 82.3	56.6	57.2	12.1	72.1	78.7	27.5	65.5	47.5
<i>Material Science & Engineering</i>	17.0	41.8	44.7	47.3	16.9	<u>63.7</u>	14.0	21.1	29.3	44.6	36.0	33.0
<i>Ecology & Environment</i>	14.6	47.4	49.1	47.8	13.7	<u>53.2</u>	13.7	36.9	36.9	49.3	43.5	42.8
<i>General Knowledge</i>	12.8	61.7	70.0	61.6	14.6	<u>73.2</u>	19.9	37.7	52.6	59.0	53.8	59.8
<i>Industry</i>	55.8	51.1	38.3	61.1	37.2	28.8	34.0	43.3	53.1	<u>62.2</u>	49.0	38.1
<i>Medicine</i>	12.2	41.2	44.9	<u>49.0</u>	11.3	59.1	9.7	20.8	29.7	32.8	34.8	25.6
<i>Units & Measurements</i>	14.2	34.7	42.3	33.1	16.6	<u>44.6</u>	13.5	26.3	29.3	30.8	26.5	23.3
<i>Chemistry</i>	39.6	44.4	37.7	50.0	29.0	46.0	22.9	38.8	43.1	<u>46.9</u>	24.7	44.0
Type Taxonomy Discovery Task												
<i>Education</i>	8.2	19.7	14.8	25.4	9.9	22.0	6.2	10.8	16.8	25.4	17.0	25.6
<i>Events</i>	5.0	14.9	32.9	19.5	5.6	● 34.4	6.3	6.5	13.3	27.5	9.6	29.1
<i>Finance</i>	8.4	20.0	▲ 42.1	17.1	6.5	◆ 33.8	7.9	9.0	16.2	21.7	10.2	26.8
<i>Food & Beverage</i>	7.4	18.6	14.7	24.0	8.8	20.0	7.0	8.1	14.3	17.7	9.3	<u>25.0</u>
<i>Agriculture</i>	1.6	5.4	<u>8.5</u>	5.1	1.7	8.4	1.8	2.1	4.1	5.7	3.1	5.9
<i>Geography</i>	11.0	19.7	19.0	25.5	12.5	24.2	4.8	14.1	26.1	<u>31.6</u>	13.6	31.3
<i>Material Science & Engineering</i>	1.9	7.5	<u>15.8</u>	7.1	2.2	14.4	2.0	2.6	7.3	9.9	3.1	11.2
<i>Biology & Life Science</i>	2.0	6.2	<u>10.8</u>	5.9	1.9	8.5	2.1	2.2	3.7	5.8	2.9	6.2
<i>Ecology & Environment</i>	1.7	6.3	<u>15.5</u>	7.1	1.9	12.9	2.0	2.3	5.4	8.4	2.4	8.7
<i>General Knowledge</i>	2.3	9.9	<u>26.7</u>	12.3	3.0	21.2	2.6	4.0	10.7	14.6	4.3	17.5
<i>Industry</i>	1.7	9.4	12.3	9.9	2.1	<u>14.7</u>	2.1	3.6	7.7	12.1	3.3	12.9
<i>Medicine</i>	2.1	9.3	<u>16.0</u>	9.7	2.2	13.7	2.3	2.7	6.6	10.3	3.8	11.2
<i>Units & Measurements</i>	3.0	10.5	<u>23.0</u>	11.2	3.1	21.5	2.0	3.6	9.2	13.4	5.3	13.0
<i>Chemistry</i>	2.1	7.2	<u>13.5</u>	6.9	2.1	12.2	2.2	2.4	6.4	7.9	3.3	8.4
Non-Taxonomic Relation Extraction Task												
<i>Education</i>	7.0	11.6	<u>25.9</u>	14.6	7.4	8.1	9.8	6.9	17.8	12.7	4.5	9.2
<i>Events</i>	0.0	40.0	42.9	26.7	26.7	37.5	46.2	<u>54.5</u>	26.7	33.3	0.0	46.2
<i>Finance</i>	22.3	10.8	22.6	34.5	13.9	<u>36.3</u>	23.6	21.6	23.9	30.1	14.0	31.5
<i>Agriculture</i>	0.2	0.4	0.4	0.4	0.2	<u>0.6</u>	0.2	0.2	0.3	0.4	0.3	0.3
<i>Material Science & Engineering</i>	6.1	7.0	<u>12.2</u>	10.5	4.6	11.6	6.1	5.5	10.3	9.4	3.1	10.1
<i>Biology & Life Science</i>	12.4	<u>25.5</u>	15.4	13.8	10.9	13.0	20.9	16.5	13.2	13.6	13.6	14.3
<i>Ecology & Environment</i>	3.1	6.7	<u>9.0</u>	7.1	2.9	9.5	2.9	3.7	6.9	5.8	2.4	7.5
<i>General Knowledge</i>	7.7	11.8	<u>12.2</u>	11.0	9.2	9.0	6.8	6.0	11.5	8.7	7.4	8.5
<i>Industry</i>	19.3	23.0	13.1	17.1	17.3	20.8	<u>31.3</u>	19.6	17.2	14.6	26.1	15.9
<i>Medicine</i>	3.0	<u>6.8</u>	5.4	6.6	2.3	5.2	4.6	3.2	4.4	4.1	5.1	4.3
<i>Units & Measurements</i>	◆ 85.7	0.0	15.4	58.8	◆ 85.7	● 90.9	50.0	66.7	15.4	50.0	▲ 95.7	73.7
<i>Chemistry</i>	4.3	<u>13.8</u>	<u>13.8</u>	9.9	3.2	9.9	4.3	4.1	8.8	12.2	3.5	6.3

retrieval nor LLM handles this reliably at scale, indicating that the retrieval ceiling is a fundamental property of the task.

Scale Helps Term Typing But Not Taxonomy Discovery. Scaling from Qwen3-0.6B to Qwen3-14B produces dramatic gains on term typing (*Finance*: 22.2% → 71.8%, *General Knowledge*: 12.8% → 70.0%) but negligible gains on taxonomy discovery (*Biology & Life Science*: 2.0% → 10.8%, *Agriculture*: 1.6% → 8.5%). Scaling improves representation and instruction following, benefiting term typing and relation extraction. Taxonomy discovery, however, requires reasoning (transitive closure, subsumption, and lattice consistency) that current models may not fully encode.

Output Discipline Matters More Than Reasoning Depth. Qwen3-4B-Instruct consistently outperforms Qwen3-4B-Thinking across domains and tasks (*Finance* term typing: 71.2% vs 25.0%; *Education* taxonomy: 25.4% vs 9.9%). The thinking variant underperforms on structured extrac-

tion, which requires schema compliance, controlled formatting, and grounding in retrieved candidates. Alignment and output discipline matter more than raw reasoning depth for OL tasks.

Domain Coverage Is the Binding Constraint for Hard Domains. *Agriculture* and *Biology & Life Science* perform poorly on both taxonomy discovery and non-taxonomic RE, with taxonomy discovery in *Biology & Life Science* reaching only 2–11% and non-taxonomic RE in *Agriculture* averaging 0.2–0.6%. Failures arise from limited domain coverage: specialized terminology (e.g., Latin species names, crop traits) and complex hierarchies (e.g. GO has 40,000+ terms) overwhelm embeddings and LLMs. In contrast, high-resource domains like *Units & Measurements* achieve 85–95% F1 on non-taxonomic RE because relations are largely memorized, making retrieval less impactful. This asymmetry extends across domains: models that excel in high-resource domains do not transfer their performance to specialized ones. For example, Qwen3-14B scores 72% in *Finance* but 45% in *Medicine* and *Units & Measurements*, while Qwen3-8B scores 82% in *Geography* but 35–41% in other domains. These results suggest that a single RAG pipeline may not generalize across the full breadth of OL domains.

Model Family Matters Independently of Scale. Gemma3-4B achieves 54.5% on *Events* non-taxonomic RE (the highest among all models) outperforming Qwen3-14B (42.9%) and Qwen3-Next-80B (37.5%). Gemma3-1B similarly exceeds much larger models on *Biology & Life Science* relation extraction (20.9% vs Qwen3-14B’s 15.4%), and on *Events* (46.2% vs Qwen3-8B’s 40.0%). This pattern reflects Gemma’s stronger calibration for relational phrase recognition [30], which is critical for span-based relation extraction, a capability that appears to arise from training alignment rather than parameter count. For non-taxonomic RE specifically, smaller Gemma models offer a substantial computational efficiency advantage over larger Qwen alternatives, underscoring that task-aware model selection outperforms defaulting to the largest available model.

Memorization vs. Structural Alignment. To verify whether strong LLM performance in resource-rich domains reflects memorization, we conducted perplexity-based contamination analysis [28] on 1,191 relation pairs across high-resource domains (i.e., *Finance* and *Units & Measurements*) using the Qwen family of LLMs (details in supplementary materials section 3). *Units & Measurements* ontologies (OM [59] and QUDT [31]) show consistently low to moderate perplexity. In particular, OM exhibits low perplexity (mean: 128.7–251.6, with 29–61% of samples below 100), providing strong evidence of pretraining exposure and thus a high likelihood of dataset contamination. While QUDT shows higher but decreasing perplexity with model scale (mean: 177.5–1133.6), indicating partial familiarity, but it is less consistently memorized than OM. In contrast, the *Finance* domain (GoodRelations [48]) exhibits extremely high perplexity (mean > 6,148, with ~93% of samples exceeding 400) despite achieving 71.8–82.7% F1 on term typing, demonstrating that strong performance can occur without direct memorization. This dissociation suggests that benchmark performance reflects structural task-domain compatibility rather than simple data memorization.

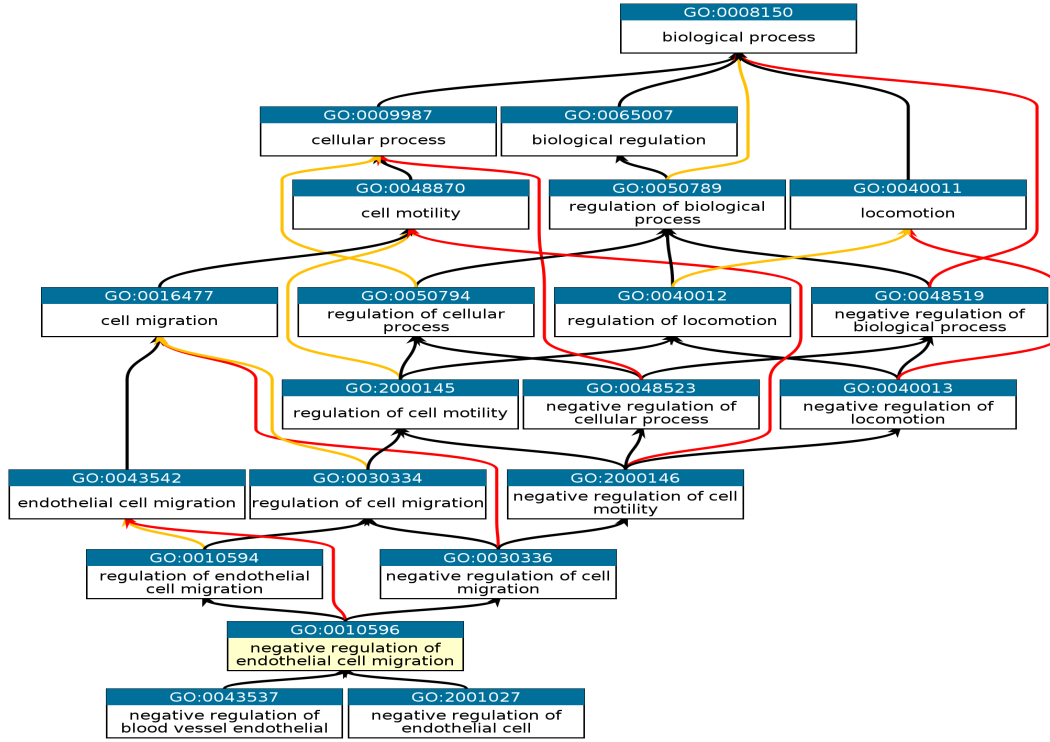
These results deepen the pattern established in the retriever analysis: *performance in OL is determined by the alignment between task structure, domain characteristics, and model training—not by scale or architecture alone*. Taxonomy discovery resists improvements in both retrieval and generation because it requires reasoning that current models do not natively encode in the real-world search space. RAG adds the most value when models are partially informed, a pattern revealed only by benchmarking across diverse domains. Instruction tuning generally outperforms reasoning variants because structured extraction demands output discipline, not open-ended inference. Domain coverage constrains the hardest tasks more than model capability does.

2.4 Analysis

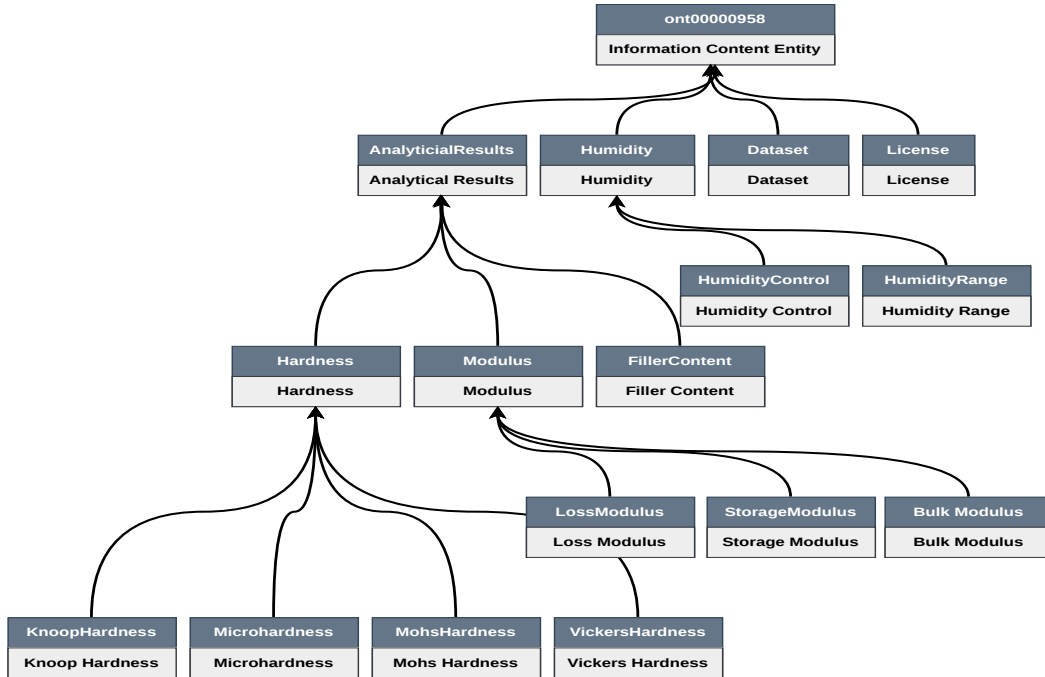
To understand OL task behavior beyond aggregate metrics, we analyze two ontologies with contrasting scale and structure: *Gene Ontology (GO)*⁶ [2, 20], a large hierarchical ontology with $Cscore = 0.71$, and *Materials and Data Science (MDSOnto)*⁷ [91], a mid-sized ontology with $Cscore = 0.38$. These provide a controlled setting to examine whether failures arise from model limitations or ontological structure.

⁶https://ontolearner.readthedocs.io/benchmarking/biology_and_life_sciences/go.html

⁷https://ontolearner.readthedocs.io/benchmarking/materials_science_and_engineering/mdsonto.html



(a) Gene Ontology is a large hierarchical ontology with 62,046 classes, 156,430 taxonomic relations. This chart presents ancestor term GO:0010596 (*negative regulation of endothelial cell migration*). **Black** arrows indicate *is-a* relationships, **Red** arrows denote the non-taxonomic relationship *negatively regulates*, and **Orange** arrows denote the non-taxonomic relationship *regulates*. The diagram was obtained from QuickGO (specifically from <https://www.ebi.ac.uk/QuickGO/term/GO:0010596>).

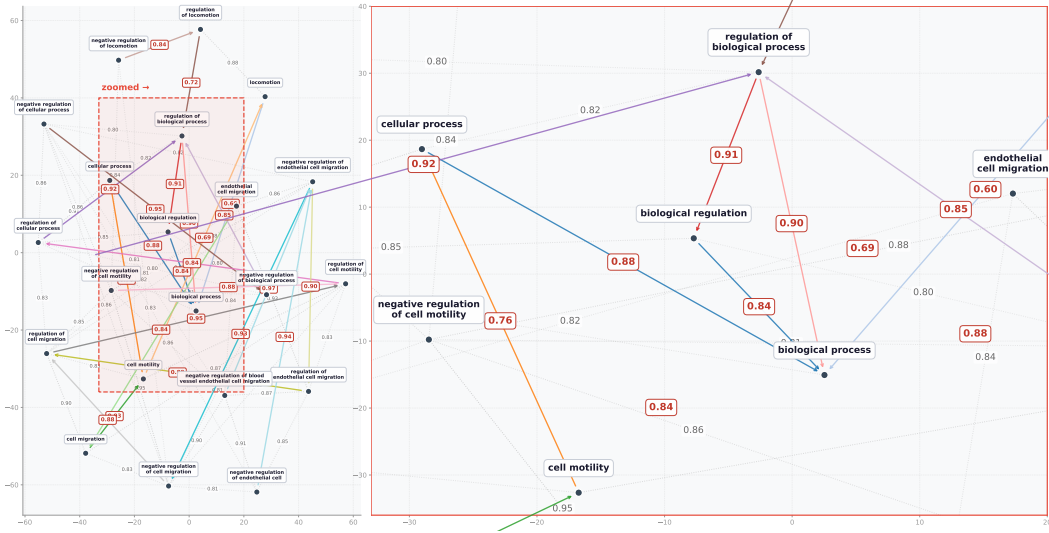


(b) Ancestor chart for the Material Data Science Ontology. **Black** arrows indicate *is-a* relationships.

Figure 3: Illustrations of smoke tests.

(a) Recall@15 for different retrieval models across taxonomy discovery and non-taxonomic RE tasks for the GO and MDSOnto.

	Qwen3-8B**	Qwen3-4B**	Qwen3-0.6B**	Qwen3-8B	MPNET**	Qwen3-4B	MiniLM-L6**	Qwen3-0.6B	Nomic-MoE**	MPNET	MiniLM-L6	Nomic**	Nomic-MoE	Nomic
Gene Ontology (GO)														
Taxonomy Discovery	31	31	30	23	30	23	30	22	30	20	20	31	22	23
Non-Taxonomic RE	97	97	97	97	90	97	90	97	83	90	90	77	83	77
Average	64	64	63	60	60	60	60	59	57	55	55	54	52	50
Material Data Science Ontology (MDSOnto)														
Taxonomy Discovery	46	47	47	26	45	26	44	24	46	22	21	43	26	22
Non-Taxonomic RE	42	25	12	42	13	25	13	12	16	13	13	28	16	28
Average	44	36	29	34	29	25	28	18	31	17	17	35	21	25



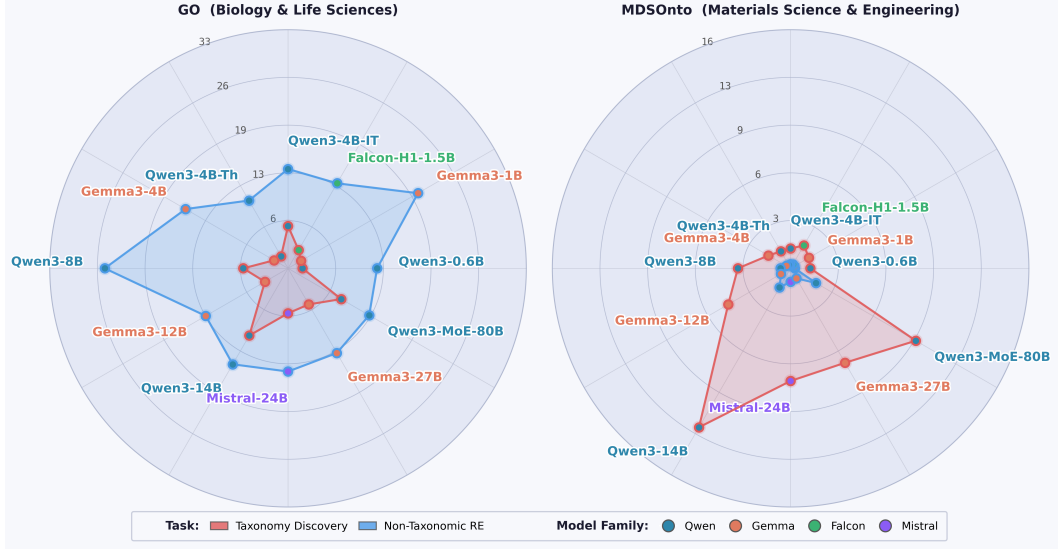
(b) t-SNE visualization of selected smoke-test taxonomic pairs from GO, generated using the Qwen3-Embedding-8B model. The plot illustrates crowded clustering in the embedding space, which contributes to lower performance in the RAG retrieval module. Colored lines indicate taxonomic relations, while dotted lines represent incorrect relations with similarity of higher than 80. The red box highlights a dense region shown as a zoomed view.

Figure 4: Retriever Limitations and Failure Modes.

2.4.1 Retriever Limitations and Failure Modes

Figure 4a shows *Recall@15* across retrieval models for both ontologies. Despite improvements from LLM-Augmented variants, recall on taxonomy discovery remains low for both ontologies (see Figure 4a), motivating a closer examination of why embeddings fail to separate hierarchical relations. To understand why embeddings fail to separate taxonomic relations clearly, we designed a smoke test on a carefully selected GO subset forming a multi-level hierarchy across several depth levels (see Figure 3a), and visualized the embedding space using t-SNE with *Qwen3-Embedding-8B* (see Figure 4b). Three failure modes emerge consistently.

First, embedding representations collapse under dense compositional labels. Concepts such as “regulation of cell migration”, “cell migration”, and “regulation of cell motility” occupy nearly identical regions despite representing distinct hierarchy levels. This collapse is particularly pronounced in GO, where combination labels (e.g., “regulation”, “negative regulation”, “cell”, “migration”) produce semantically saturated representations with minimal vector encodings. Second, similarity is distorted by hubness [77]. High-similarity links (>0.8) appear between unrelated concepts (dotted lines in Figure 4b), while genuine parent-child pairs span a wide overlapping range (0.60–0.97). For instance, “negative regulation of endothelial cell migration” shows 0.86 similarity to “regulation



(a) F1 scores of LLMs on taxonomy discovery and non-taxonomic RE across GO and MDSOnto ontologies.

(b) Model performance on GO and MDSOnto ontologies. Accuracies shown as percentages for 393 samples from GO – 260 for taxonomy discovery + 133 for non-taxonomic re tasks – and 223 samples from taxonomy discovery tasks using MDSOnto. Entropy reported as mean $\pm$ std. Δ_{calib} = entropy difference (correct – incorrect); positive values indicate good calibration. FP/FN = False Positive/Negative counts. Hall = hallucinated samples.

Model	F1-Score (%)			Attention		Calibration		Error Bias				Error Types	
	GO _T	GO _R	MDS _T	GO _{entropy}	MDS _{entropy}	GO Δ_{calib}	MDS Δ_{calib}	GO _{FP}	GO _{FN}	MDS _{FP}	MDS _{FN}	GO _{Hall}	MDS _{Hall}
Qwen3-0.6B	41.9	16.7	39.7	0.67 $\pm$ 0.03	0.69 $\pm$ 0.01	0.024	0.005	201	0	167	0	108	75
Gemma-3-1b-it	0.0	10.5	3.8	2.56 $\pm$ 0.14	2.63 $\pm$ 0.02	-0.163	0.014	12	75	0	51	8	0
Falcon-H1-1.5B-Deep	64.7	13.2	27.9	1.75 $\pm$ 0.08	1.80 $\pm$ 0.03	-0.110	-1.805	127	10	14	43	97	3
Gemma-3-4b-it	44.1	17.3	52.3	1.78 $\pm$ 0.08	1.82 $\pm$ 0.01	-0.026	0.006	238	4	58	15	142	16
Qwen3-4B-Instruct-2507	73.0	18.5	52.3	1.12 $\pm$ 0.07	1.16 $\pm$ 0.01	0.072	0.002	141	4	10	32	116	2
Qwen3-4B-Thinking-2507	0.0	0.0	0.0	1.09 $\pm$ 0.05	1.10 $\pm$ 0.01	-1.088	-1.098	0	1	0	3	0	0
Qwen3-8B	73.0	16.1	51.2	1.53 $\pm$ 0.08	1.58 $\pm$ 0.01	0.101	0.003	159	4	6	34	139	1
Gemma-3-12b-it	61.9	16.2	35.0	1.37 $\pm$ 0.11	1.42 $\pm$ 0.01	0.117	0.001	199	0	11	41	158	0
Qwen3-14B	78.4	17.2	22.2	1.44 $\pm$ 0.08	1.48 $\pm$ 0.01	0.056	0.002	130	9	2	47	116	0
Mistral-Small-3.2-24B	75.1	15.9	49.4	1.83 $\pm$ 0.11	1.86 $\pm$ 0.01	0.152	0.008	155	4	3	36	132	0
Gemma-3-27b-it	71.9	15.5	51.8	0.98 $\pm$ 0.05	0.99 $\pm$ 0.01	0.065	-0.000	174	0	5	34	146	0
Qwen3-Next-80B-A3B	73.9	16.2	34.3	2.71 $\pm$ 0.11	2.77 $\pm$ 0.01	0.143	0.003	156	4	3	43	132	0

Figure 5: LLM Behavior Analysis.

of locomotion” despite no taxonomic relation. This overlap makes threshold-based retrieval inherently unreliable. Third, embeddings saturate at a semantic ceiling where relatedness is preserved, but taxonomic entailment is lost. The model captures that “cell migration”, “cell motility”, and “locomotion” are related, but cannot encode the directional specificity required for *is-a*. Structurally distinct pairs such as [“biological process”, “regulation of biological process”] with similarity 0.91 and [“cellular process”, “regulation of cellular process”] with similarity 0.88 produce near-identical similarity patterns.

Collectively, these behaviors point to a core technical limitation of embedding anisotropy [66, 93], where representations occupy a narrow cone in vector space, causing cosine similarities to cluster in a poorly discriminative range for *is-a* relations. Critically, the severity of this effect is determined by the ontology properties (i.e., label compositionality, class density, and relational complexity) rather than model scale. Embeddings encode relatedness, not entailment, and scaling does not resolve this mismatch when the ontology amplifies it.

2.4.2 LLM Behavior Analysis

Figure 5b and Figure 5a reveal three behavioral patterns through attention, calibration, error bias, and hallucination analysis. 1) *Task asymmetry*. In controlled smoke tests, models achieve high F1 in taxonomy discovery (≈ 70 – 75%) but collapse to ≈ 15 – 18% in non-taxonomic RE (e.g., Qwen3-8B: 73.0% vs 16.1%; Mistral-24B: 75.1% vs 15.9%). This exposes a fundamental weakness in non-hierarchical reasoning. Qwen3-4B-Thinking reaches 0.0% F1, showing that open-ended chain-of-thought reasoning conflicts with the strict schema compliance required for structured OL extraction.

Reasoning depth does not substitute for output discipline. 2) *Scaling helps selectively*. Despite a $\approx 4\times$ increase in entropy from smaller to larger models (Qwen3-0.6B: 0.67 $\rightarrow$ Qwen3-Next-80B: 2.71), only marginal improvements appear in relational reasoning (Gemma-3-1B: 10.5% $\rightarrow$ Gemma-3-27B: 15.5% on GO non-taxonomic RE). Therefore, entropy reflects the sample’s structural complexity rather than the model’s capability, and it is not a reliable predictor of task performance. 3) *Model behavior diverges by ontology*. Calibration shifts dramatically across ontologies for identical models (e.g., $\Delta_{\text{calib}} = -1.088$ and -1.805), indicating that even when models predict incorrectly, predictions are made with high confidence. Calibration is therefore not an intrinsic model property but a *model $\times$ domain* interaction.

Error bias analysis reveals a structural effect. The FP/FN ratio reverses between ontologies, with systematic over-prediction on GO (Qwen3-0.6B: 201 FP vs 0 FN; Gemma-12B: 199 FP vs 0 FN; Qwen3-8B: 159 FP vs 4 FN), and under-prediction on MDSOnto (Gemma-1B: 0 FP vs 51 FN; Qwen3-Next-80B: 3 FP vs 43 FN). Hallucination follows the same pattern; GO produces up to 158 hallucinations per model while MDSOnto produces 0–3. These reversals correlate with ontological properties such as class distribution, relational density, and hierarchy, not architecture or scale.

These results converge on a single structural conclusion: *in OL, failure modes are ontology-determined rather than model-determined*. Embedding anisotropy, calibration collapse, error bias reversal, and hallucination concentration scale with ontological complexity rather than model size.

3 Discussion

That failure in OL is ontology-determined rather than model-determined, mirroring the fragmentation identified across three decades of OL research, in which each generation optimized for narrow settings while the broader challenge remained unsolved. Therefore, the path forward requires not just better models but also ontology-aware architectures, richer evaluation frameworks, and infrastructure capable of systematically exposing these structural effects.

Overcoming the retrieval ceiling requires moving beyond single-architecture solutions. The smoke test analysis points to three concrete directions. First, contrastive learning objectives [35] designed to separate entailment from relatedness would directly address embedding collapse and saturation. Second, hyperbolic embedding spaces [104] are well-suited to hierarchical structures and could alleviate the hubness effect by explicitly representing taxonomic depth. Third, anisotropy mitigation techniques, such as whitening transformations, address the geometric crowding that makes cosine similarity unreliable for *is-a*-typed pairs. Hybrid architectures that combine these approaches with symbolic features represent the most promising near-term direction for taxonomy discovery.

Current OL evaluation practice relies almost exclusively on F1-score, which is insufficient to characterize the failure modes this benchmark exposes. Four findings demand broader evaluation criteria. First, contamination analysis demonstrates that strong performance can occur both with and without pretraining exposure: *Finance* achieves high F1 (71.8–82.7%) despite extreme perplexity (mean 6,148), while *Units & Measurements* achieves similar F1 with low perplexity (mean 128.7–251.6), confirming that benchmark performance reflects structural task-domain compatibility rather than simply data leakage—a distinction F1 alone cannot reveal. Second, the error-bias reversal (extreme overprediction on GO, systematic underprediction on MDSOnto) is invisible to F1 but reveals that the model’s behavior is governed by ontological structure rather than learned relational understanding. Third, the calibration analysis shows that models make confident predictions regardless of correctness and that calibration is a *model $\times$ domain* interaction rather than an intrinsic model property, with the distinction only visible through cross-ontology evaluation. Fourth, the output discipline finding establishes that instruction-tuned models systematically outperform reasoning-optimized variants on structured extraction, suggesting that model selection for OL workflows should prioritize alignment and formatting discipline over raw reasoning capability. A single RAG pipeline optimized for high-resource domains will not generalize to specialized ones. Domain-specific retriever fine-tuning or ontology-aware pipeline design is required for consistently strong performance across the full breadth of OL.

The convergence of ontologies across 22 domains, three tasks, and more than 30 models points to a conclusion the field has approached but never been able to state precisely: the bottleneck in OL is not model capability but the structural mismatch between how models encode knowledge and how ontologies organize it. Closing this gap requires infrastructure that makes the mismatch visi-

ble, reproducible, and comparable across methods; precisely what has been missing. OntoLearner provides that infrastructure, 180 ontologies spanning the full complexity spectrum, standardized evaluation across retrieval and LLMs, and the capacity to evaluate and compare hybrid approaches reproducibly, making findings from approaches like LLM-Augmented retrieval reproducible rather than observational. Our commitment to advancing this agenda is reflected in the LLMs4OL challenges at ISWC [38, 7], which establish shared tasks and community benchmarks for reproducible OL research. More broadly, OntoLearner aims to support a new generation of hybrid knowledge-engineering workflows in which human expertise, curated ontologies, and foundation models collaborate—not sequentially, but as components of a unified, evaluable system.

4 Methods

The development of OntoLearner was guided by a systematic requirements analysis of existing ontology engineering tools (see Table 1a), which revealed key gaps, including limited cross-domain support, minimal LLM integration, and weak modularity and reusability. Insights from our prior empirical study on LLMs for OL [8] and lessons learned from the LLMs4OL challenge series organization [38, 39] shaped the framework’s requirements. This section outlines the requirement specification, then describes the software implementation and architectural compliance, and finally details the experimental setup for LLM analysis in OL.

4.1 Requirements Specification

Requirements guide the systematic development of a robust and high-quality library. The MASTER template by Chris Rupp and The SOPHISTS [96, 110] is used to specify both functional and non-functional requirements in a clear, consistent, complete, and testable manner. It employs the keywords “shall,” “should,” and “will” to indicate obligation levels: “shall” for mandatory requirements, “should” for recommended but optional ones, and “will” for planned future enhancements.

The documented requirements are listed in Table 3. Functional requirements (labeled “F”) define the core capabilities of the OntoLearner Python library, comprising 14 items. These cover ontology management (F1–F4) for hosting and accessing cross-domain ontologies, ontology exploration (F5–F7), core OL tasks (F8–F11), learning models for LLM integration (F12–F13), and user customizations and interactions (F14). Moreover, non-functional requirements (denoted by “N”) focus on quality aspects such as performance, usability, compatibility, and maintainability, which include 8 requirements, each defining the qualities of the framework, including performance (N1, N2, N3), usability (N4, N5), scalability (N6, N7), and maintainability (N8). Overall, the requirements are designed to encompass multiple objectives, as highlighted in Table 1a.

The *Proven Feature* (PF) column in Table 3 lists implemented and verified features, while the *Current Deficit* (CD) column highlights areas that need further development. This clear separation between completed features and identified gaps facilitates effective tracking of development progress and helps prioritize future improvements.

The OntoLearner library satisfies 13 of 16 functional and 8 of 9 non-functional requirements, reflecting a mature and practically usable system while leaving scope for further development. This high adherence demonstrates the tool’s maturity and its suitability for practical applications in OL and knowledge representation, while also indicating areas for further refinement to ensure continuous improvement. Nevertheless, the requirements were defined by the main developers based on the library’s design objectives and architecture decisions; thus, no formal agreement measures were applied. This approach is consistent with research-oriented software artifacts, in which requirements serve as a structured specification rather than the outcome of multi-stakeholder negotiation. To mitigate over-specialization and support future development, several requirements are intentionally abstract (e.g., F12–F13, N8) to support extensibility and future integration of additional AI components and learning interaction paradigms.

4.2 Software Implementation

OntoLearner is a modular Python framework for OL and reuse. It comprises three core components—Ontologizers, Learning Tasks, and Learner Models—plus an ontology complexity scorer

Table 3: Functional (F) and Non-Functional (N) requirements for the OntoLearner Python library, specified using the MASTER requirements template by Rupp and The SOPHISTS [96, 110]. The *Proven Feature* (PF) column lists implemented functionalities; the *Current Deficit* (CD) column indicates areas for future development.

	Requirement	PF	CD
F1	The library shall be able to load ontologies from various local file formats (e.g., OWL, XML, RDF, TTL).	✓	
F2	The library shall provide a Python import module to load ontologies and datasets automatically.	✓	
F3	The library shall provide access to an external repository on Huggingface for loading ontologies and task datasets.	✓	
F4	The library should adapt cross-domain ontologies.	✓	
F5	The library should allow users to measure and display ontological metrics for loaded ontologies.	✓	
F6	The library should allow users to create, view, and modify ontology metadata (e.g., name, domain, version, license).	✓	
F7	The library should allow users to browse through classes and relations of given ontologies.	✓	
F8	The library shall support the term-typing task for OL.	✓	
F9	The library shall support OL’s taxonomy discovery task.	✓	
F10	The library shall support the non-taxonomic relation extraction task of OL.	✓	
F11	The library will enable the Text2Onto (terminology and type extraction) task in OL.	✓	
F12	The library shall integrate with external learners (e.g., LLMs, retrieval models).	✓	
F13	The library should enable users to customize existing learning models provided by the library, including their configuration parameters.	✓	
F14	The library should allow users to customize learning tasks, including task definitions and configurations.		✓
F15	The library should support multi-agent or ensemble learning strategies for OL tasks.		✓
F16	The library should provide a graphical user interface for learners to interactively explore and execute ontologies.		✓
N1	The library shall efficiently handle increasing amounts of data and concurrent users.	✓	
N2	The library should process large ontologies, including those with thousands of classes and properties, within a reasonable time frame.	✓	
N3	The library should optimize memory usage to efficiently manage large ontologies.	✓	
N4	The library shall be intuitive, easy to use, and extendible, with comprehensive user guides, examples, tutorials, and clear documentation.	✓	
N5	The library will handle exceptions gracefully, providing the user with meaningful error messages.	✓	
N6	The library should be compatible with major operating systems (Windows, macOS, Linux) and support multiple Python versions.	✓	
N7	The library shall be provided as a Python package and be easily installable via PyPI.	✓	
N8	The library should adhere to best practices for software development, including version control, modular design, code-level documentation, and automated testing.	✓	
N9	The library should isolate optional extensions (e.g., UI, advanced learners) from the core library to prevent coupling and regression.		✓

and the Text2Onto corpus synthesis module (see Figure 6). The framework is available on GitHub, Hugging Face, PyPI, and Read the Docs. We describe each component below.

4.2.1 Software Core Components

1. Ontologizer – *A foundational module for ontology modularization that enables specification, evaluation, documentation, reuse, and seamless Pythonic import.* As noted in the OD analysis in Related Work, domain-specific ontology repositories like BioPortal and AgroPortal promote alignment with community standards and research practices. To support similar integration across domains, the Ontologizer module offers an extensible Python interface for loading, inspecting, and repurposing ontologies within OntoLearner workflows. It transforms ontologies into programmatically accessible assets, enabling communities to adapt and reuse cross-domain ontologies (F4) as part of their workflows. This module offers the following key functionalities:

- *Processing*: To begin, inherit from the base ontology class⁸, which parses supported ontology file formats (OWL, RDF, XML, TTL) into a Pythonic object for programmatic manipulation. This ensures compatibility with Ontologizer’s core functions. If needed, predefined

⁸<https://github.com/sciknoworg/OntoLearner/blob/main/ontolearner/base/ontology.py>

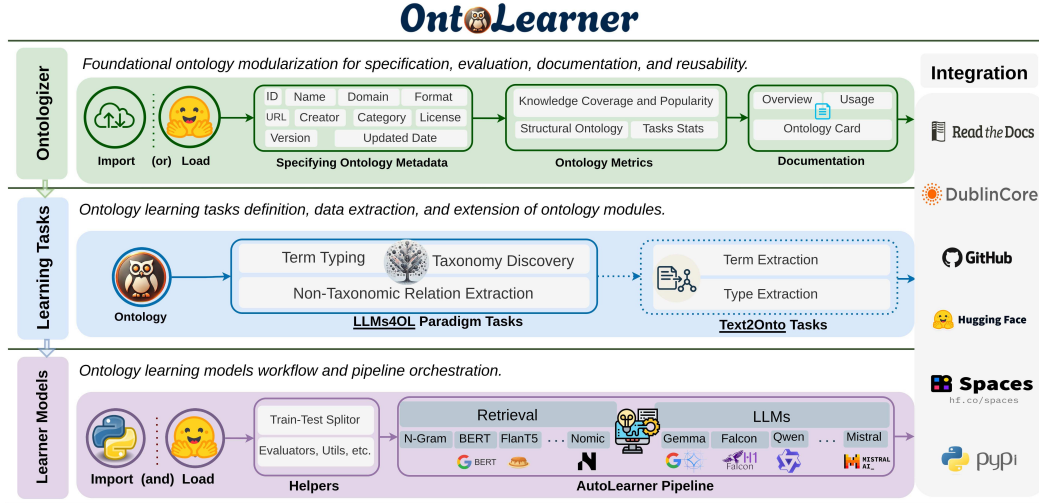


Figure 6: The conceptual and functional architecture of the OntoLearner library, illustrating its modular design for ontology access and learning within LLM workflows.

444 methods can be adjusted for specific cases—for example, ignoring blank nodes in certain
445 learning tasks⁹.

- 446 • *Import (or) Load*: Ontologies can be directly imported from the web or through Hugging
447 Face Datasets (F3), streamlining the process of obtaining ontology data without manual file
448 handling. It also supports loading data from the local machine (F1).
- 449 • *Metadata Specification*: Essential metadata attributes are defined for each ontology (i.e.,
450 ID, name, domain, format, version, creator, URL, license, category, update-date) (F6) to
451 ensure traceability, version control, and compliance with FAIR principles.
- 452 • *Ontology Metrics Evaluation*: The module supports automated measurement of key on-
453 tology metrics (F5), such as knowledge coverage (i.e., number of classes, properties, in-
454 dividuals), structural complexity (i.e., hierarchical, breadth metrics), and popularity (i.e.,
455 nodes, edges, and leaf nodes). These metrics provide insights into the ontology’s quality
456 and suitability for various tasks.
- 457 • *Automated Documentation*: Ontology documentation is generated following a Read-the-
458 Docs style, providing clear, user-friendly insights into ontology usage, coverage, and struc-
459 ture (N4).
- 460 • *Versioning*: Versioning supports the controlled evolution of ontology modules. Each on-
461 tology is assigned a version stored in its metadata, allowing users to track changes over
462 time. Updates can occur in two ways: by submitting newer versions (e.g., via Hugging
463 Face) or by proposing modifications through version control systems like GitHub (e.g.,
464 pull requests). These platforms enable collaborative development by tracking changes and
465 facilitating community-driven improvements, ensuring alignment with evolving needs.

466 Once an ontology has been processed and integrated into OntoLearner, it can be imported using
467 Python’s standard import mechanism (F2). This approach removes the need for manual downloads
468 or complex loading procedures from external sources and provides users with easy-to-use function-
469 ality (N4). Moreover, as OntoLearner evolves, the ontology module will continuously be enhanced
470 with new features to accommodate emerging requirements in ontology engineering, ensuring ongo-
471 ing compatibility and functionality (N4, N8).

472 To support concurrent user involvement, this component maintains strict isolation between users and
473 ontologies. The OL enables multiple simultaneous usage and updates, while subsequent modules
474 rely solely on modularized ontologies for extension or learning strategies (N1). This design adheres

⁹Example fix: https://github.com/sciknoworg/OntoLearner/blob/main/ontolearner/ontology/material_science_engineering.py#L48

to software engineering best practices (N8) and promotes scalable cross-domain ontology enrichment within OntoLearner. For example, we utilize multi-processing capabilities (N3) to optimize memory usage and efficiently manage large ontologies, even those with thousands of classes and properties (N2).

2. Learning Tasks – *The second step after Ontologizer is defining OL tasks for data curation.*

Within the OntoLearner framework, the modularized ontologies are extended with OL capabilities that automate core learning tasks. This extension is guided under the three core OL tasks we introduced via the LLMs4OL paradigm in 2023 [8], which leverage LLMs to automate the key OL process (F8, F9, F10). The LLMs4OL paradigm is structured around three primary tasks essential for developing a primitive ontology: 1) Term Typing, the discovery of generalized type for a lexical term, 2) Taxonomy Discovery, the identification of the taxonomic hierarchy (is-a or subclass relationships) between type pairs, and 3) Non-Taxonomic Relationship Extraction, the extraction of non-taxonomic, semantic relations between types (relationships beyond “is-a”). By incorporating these tasks, OntoLearner ensures that its modularized ontologies, created with Ontologizer, are not only structured and reusable but also capable of continuous learning via automated OL techniques.

Additionally, OntoLearner incorporates a preliminary implementation of the Text2Onto module [19], which focuses on extracting ontological terms and types directly from raw text (F11). This component (described in more detail in subsection 4.2.2) is currently under active development and leverages LLMs for synthetic data generation to support term extraction and term typing, extending the LLMs4OL paradigm and enriching the framework’s learning objectives also to derive structured knowledge from unstructured sources. Notably, Text2Onto is designed to function independently of the LLMs4OL pipeline, ensuring modularity and preserving the integrity of existing components (N8). Users can import or load LLMs4OL tasks as inputs to Text2Onto, enabling flexible and extensible data extraction workflows for OL.

3. Learner Models – *Enables workflow and pipeline orchestration of LLMs for OL, supporting seamless model execution and result exploration.*

The Learner Models module serves as the orchestration layer of OntoLearner for managing OL workflows (F12, F13). It provides a flexible, extensible environment for designing, configuring, and experimenting with LLM-driven OL models. Key components include:

- *Machine Learning Utilities*: Core tools for robust modeling, such as train-test splits, label mappings, and evaluation metrics, are provided and easily importable (F12, N8).
 - `train_test_split`. The train-test split procedure is specifically designed to prevent data leakage across all OL tasks. Unlike a naive random split (e.g., directly applying Scikit-Learn `train_test_split` [87]), OntoLearner implements a multi-stage, leakage-aware splitting algorithm including: 1) *Term-Level Stratification* stage, where term types are first split using a stratified grouping strategy based on the least frequent associated type of each term. This balances rare and frequent types across splits. 2) *Taxonomy Leakage Prevention* stage, in which taxonomic relations are then partitioned while enforcing that no term appearing in the training set is introduced into the test set. For relations not constrained by prior term assignment, the implementation leverages Scikit-Learn `train_test_split` to ensure statistically robust random sampling with reproducibility guarantees. 3) *Relation-Type Preservation* stage, where non-taxonomic relations are split proportionally within each relation category to preserve semantic distribution across splits.
 - `LabelMapper`. A lightweight post-processing component that maps free-form LLM outputs to a fixed set of valid task labels. It was introduced because LLMs often generate lexical variations or loosely formatted answers (e.g., “true”, “Yes.”, “It is correct”), which can break evaluation pipelines. By normalizing such outputs into canonical labels, `LabelMapper` ensures consistent, valid, and evaluation-ready predictions in OL tasks.
 - `evaluation_report`. A task-aware evaluation component that computes precision, recall, and F1-score for OL tasks using normalized pair- and triple-level matching. It ensures fair, consistent, and reproducible performance assessment, including proper handling of symmetric relations and lexical variations when required.

Algorithm 1 Text2Onto Synthetic Corpus Generation

Require: Ontology $\mathcal{O}$, Document size parameter k , LLM configuration θ

Ensure: Synthetic corpus $\mathcal{C}$

- 1: Extract terms T , types Y , taxonomic relations R from $\mathcal{O}$
 - 2: Partition (T, Y) into subsets $\{P_1, P_2, \dots, P_n\}$ using CMST-inspired algorithm with capacity k
 - 3: **for** each partition P_i **do**
 - 4: Generate CNL statements S_i from P_i
 - 5: Paraphrase S_i using LLM with configuration θ to obtain document d_i
 - 6: Annotate d_i with ground truth terms and types
 - 7: **end for**
 - 8: Aggregate all d_i into corpus $\mathcal{C}$
 - 9: Split $\mathcal{C}$ into train, development, and test sets
-

- *Learner Pipeline*: This component orchestrates learner models into three primary categories: AutoRetriever, AutoLLM, and AutoLearner, following a modular and extensible design pattern (N8).

- AutoRetriever. This module plays a standard interface role for retrieval systems and supports a range of retrieval approaches, from traditional (e.g., NGrams) to advanced neural embeddings (e.g., BERT [26], Nomic-embed [82]), and enables search and navigation across ontology classes (F7).
- AutoLLM. This module abstracts over a variety of LLMs—from compact models like Qwen2.5-0.5B [109] to larger ones such as LLaMA [112] and DeepSeek [64]—for ontology-related tasks. It provides a uniform interface for LLM-based operations on ontology tasks.
- AutoLearner. This module acts as the core abstraction layer, defining a common interface for adapting diverse learner models. The AutoRetriever and AutoLLM components are intentionally designed to keep retrieval and LLM logic separate from learner models, following the Interface Segregation Principle (ISP) [79]. This separation allows any learner to use a retriever or LLM backend provided to it, regardless of the internal implementation. Such a design enables seamless integration and switching between different modeling techniques without requiring re-implementation of existing learner logic. It also supports the Loose Coupling principle [85], ensuring that learners do not need to know how the LLM or retriever is instantiated or managed internally. In line with extensibility principles [76], AutoLearner defines learning strategies that can produce specialized modules such as AutoRetrieverLearner (when an AutoRetriever is provided) and AutoLLMLearner (when an AutoLLM is provided). Furthermore, AutoLLM and AutoRetriever can be combined in a retrieval-augmented generation (RAG) [63, 36] setup through the AutoRAGLearner module. Overall, AutoLearner supports composition by allowing other learner categories to be integrated as long as they follow the specified interface or signature (F12, F14, N4, N8).

To enhance prompt management in LLM workflows, OntoLearner includes an additional submodule, AutoPrompt, that allows users to define custom prompts (F14). If no prompt is provided, a predefined default is used. For instance, a user may combine a sentence-transformer model [94] from AutoRetriever with a smaller LLM like Qwen2.5-0.5B [109] in a RAG configuration using a custom prompt. OntoLearner thus provides a user-friendly yet powerful environment for diverse learning scenarios (F14, N8).

4.2.2 Synthetic Corpus Generation

The Text2Onto module is a key extension of OntoLearner. While the broader goal of OL is to generate ontologies from unstructured text, the absence of gold-standard corpora poses a significant challenge for benchmarking such pipelines. To bridge this gap, Text2Onto focuses on synthetic corpus construction—generating unstructured text aligned with existing ontologies—to enable controlled evaluation of ontology generation workflows. The module leverages recent advances in LLM-based synthetic data generation (SDG) [55, 122] to produce documents annotated with ground-truth terms and types. The roadmap includes integrating multiple SDG algorithms that can simulate differ-

Table 4: Ontology complexity scoring framework and robustness analysis.

(a) Interpretation of the ontology complexity score $\mathcal{C}_{score}$.

Value	Interpretation
$0 < \mathcal{C}_{score} < 0.20$	Low-complexity ontology. Modest graph size, shallow depth, limited relations, and simple modelling patterns with little semantic richness.
$0.20 \leq \mathcal{C}_{score} < 0.40$	Moderate-low complexity. Contains meaningful class coverage, some hierarchy development, and moderate breadth or relational structure.
$0.40 \leq \mathcal{C}_{score} < 0.60$	Medium complexity. Ontology exhibits developed hierarchy, moderate-to-large graph structure, and richer semantic or relational diversity.
$0.60 \leq \mathcal{C}_{score} < 0.80$	High complexity. Ontology is large, multi-layered, with substantial branching, semantic features, and diverse relationships. Indicates significant modeling depth.
$\mathcal{C}_{score} \geq 0.80$	Very high complexity. Ontology is extremely large-scale or densely structured, with deep hierarchies, high fan-out, and rich semantic or lexical content. Typically challenging for reasoning, maintenance, or LLM-based processing.

(b) Robustness analysis of the ontology complexity score under different configurations. Rank stability is measured using Spearman’s ρ with respect to the original configuration. The columns correspond to: *Weight Perturbations* is $\pm 10\%$ and $\pm 20\%$ perturbations of metric family (graph, coverage, hierarchy, breadth, and dataset) weights; *Sigmoid Variation*, the variation of parameters $a \in [0.3, 0.5]$ and $b \in [5.5, 6.5]$; *Normalizations*, the Min–Max and Z-score are alternative normalization schemes; and *Uniform Weighting* is an equal weighting across all metric families.

	Weight Perturbations	Sigmoid Variation	Normalizations	Uniform Weighting
<i>Variants</i>	24	20	2	1
$\rho_{\min}$ (<i>min</i> ρ)	0.9990	1.000	0.8557	0.9968
$\bar{\rho}$ (<i>mean</i> ρ)	0.9996	1.000	0.8564	0.9968

ent characteristics of real-world text, such as homogeneous passages focused on a single type or heterogeneous passages spanning multiple types. The current implementation supports one such algorithm, detailed in Algorithm 1, based on the following workflow:

1. *Ontology Data Extraction.* Utilizing OntoLearner’s APIs, Text2Onto retrieves ontology terms (individuals), types, taxonomic relations, and non-taxonomic relations, serving as input for subsequent processes.
2. *Term and Type Partitioning.* To control document size and ensure semantic coherence, the module partitions ontology data using an algorithm inspired by the Capacitated Minimum Spanning Tree Problem (CMST) [40]. This method groups closely related terms and types into connected subgraphs, promoting homogeneity within each generated document.
3. *Synthetic Text Generation.* The module employs a two-step approach inspired by [122]: first, ontology axioms are verbalized using Controlled Natural Language (CNL) [62] based on predefined templates aligned with specific axiom patterns. For example, the axiom `:Capricciosa rdf:type :NamedPizza` is rendered as “*Capricciosa is a type of Named Pizza*”, using a template for the `rdf:type` relation. In the second step, the resulting CNL statements are paraphrased by an LLM to produce more natural, fluent, and contextually coherent text. For instance, a set of axioms about pizzas may be transformed into: “*Capricciosa, Four Seasons, and Napoletana are distinct varieties of pizza, each offering a unique combination of toppings balancing savoury and briny. All three share toppings such as anchovies, mozzarella and olives*”.

Users can tune LLM hyperparameters (e.g., model, decoding, temperature, and token limit) to control text style and diversity. Outputs can be evaluated using criteria such as coherence, readability, informativeness, and conciseness [6], and are partitioned into train and test splits for downstream tasks.

4.2.3 Ontology Complexity Scorer

Complexity Scorer. As shown in Figure 6, per ontology, Ontologizer computes key ontology metrics, including graph metrics $\mathcal{G}$ (i.e., total nodes, edges, root nodes, and leaf nodes), knowledge coverage $\mathcal{C}$ (i.e., classes, individuals, properties), hierarchical structure $\mathcal{H}$ (i.e., maximum, minimum, and average depth), breadth $\mathcal{B}$ (i.e., maximum, minimum, and average breadth), and dataset

characteristics $\mathcal{D}$ (i.e., number of term types, taxonomic/non-taxonomic relations, and average terms per type). To summarize these high-dimensional metrics, we introduced a complexity score: a single, reproducible value summarizing an ontology’s structural complexity. Rather than a standalone metric, complexity is treated as an emergent property arising from the interaction of multiple internal characteristics. This unified, absolute score relies solely on an ontology’s intrinsic properties, ensuring a self-contained measure that facilitates consistent benchmarking without external dependencies. Formally, we define this score by first representing the ontology metrics as a vector representation of $x = \{x_j | j \in \mathcal{M}\}$, where $\mathcal{M}$ is the set of all metrics: $\mathcal{M} = \{\mathcal{G} \cup \mathcal{C} \cup \mathcal{H} \cup \mathcal{B} \cup \mathcal{D}\}$. The calculation of this complexity score involves the following key steps:

1. *Metric-level Normalization.* To obtain an absolute, ontology-intrinsic measure, we first apply a monotone, sublinear transformation to each metric $\mathcal{M}$. The transformation is done by using the natural logarithm of $\tilde{x}_j = \log(1 + x_j)$. The $\log(1 + x)$ transform counts that span in high order of magnitude while preserving ordering and sensitivity at small scales. This has proven to be a logical decision for normalization due to the fact that all the metrics are mostly count-based metrics. Moreover, unlike the min-max or percentile normalization, the $\log(1 + x)$ is local to the ontology and therefore preserves the requirement that the complexity score be self-explanatory and reproducible.
2. *Weighted Aggregation.* Normalized metric values are combined with metric weights w_j that encode the relative importance of different metric families (graph, coverage, hierarchy, breadth, and dataset). The raw aggregated score $\mathcal{C}_w$ is defined as following: $\mathcal{C}_w(\mathcal{M}) = \frac{\sum_{j=1}^m w_j \tilde{x}_j}{\sum_{j=1}^m w_j} \in [0, 1]$, where the w_j is a per-metric weight that are defined as follows:

$$w_j = \begin{cases} 0.30, & \text{if } j \in \mathcal{G} \text{ (Graph)} \\ 0.25, & \text{if } j \in \mathcal{C} \text{ (Knowledge Coverage)} \\ 0.10, & \text{if } j \in \mathcal{H} \text{ (Hierarchy Structure)} \\ 0.20, & \text{if } j \in \mathcal{B} \text{ (Breadth)} \\ 0.15, & \text{if } j \in \mathcal{D} \text{ (Dataset Characteristics)} \end{cases}$$

3. *Sigmoid Squashing to $[0, 1]$.* The weighted aggregation score of $\mathcal{C}_w$ is unbounded, because $\log(1 + x)$ grows without bound. To produce a final complexity score in the range of $[0, 1]$ that is smooth and interpretable, we applied a logistic (sigmoid) mapping with fixed parameter a (steepness) and b (midpoint):

$$\mathcal{C}_{score} = \sigma_{a,b}(\mathcal{C}_w(\mathcal{M})) = \frac{1}{1 + e^{(-a(\mathcal{C}_w(\mathcal{M}) - b))}}$$

The typical parameter choices used in this work are $a = 0.4$, $b = 6.0$ (these are fixed, domain-informed constants and do not depend on the dataset). This pipeline of $x_j \rightarrow \log(1 + x_j) \rightarrow \mathcal{C}_w(\mathcal{M}) \rightarrow \sigma_{a,b}$ was chosen because it (i) compresses heavy-tailed count metrics while preserving order, (ii) uses only ontology local transformations, so scores are reproducible and most importantly independent, and (iii) produces a smooth bounded complexity score $\mathcal{C}_{score} \in [0, 1]$ whose mid-point and sensitivity can be set by interpretable parameters b and a . Table 4a represents the $\mathcal{C}_{score}$ interpretation in five scales.

Complexity Scorer Robustness Analysis. To evaluate the robustness of the proposed ontology complexity score, we perform four complementary sensitivity assessments (see Table 4b) using 180 ontologies. First, we test *weight perturbations* by varying each metric-family weight (graph, coverage, hierarchy, breadth, dataset) by $\pm 10\%$ and $\pm 20\%$ while re-normalizing the remaining weights. Second, we analyze *sigmoid variation* by changing the bounding parameters over $a \in [0.3, 0.5]$ and $b \in [5.5, 6.5]$. Third, we compare *normalization schemes* by replacing the default $\log(1 + x)$ transformation with Min-Max and Z-score normalization. Fourth, we evaluate a *uniform weighting* baseline where all metric families are assigned equal weight. The results reveal a consistent and interpretable pattern.

The robustness analysis shows that weight perturbations have a negligible impact on ranking ($\rho_{\min} = 0.9990$, $\bar{\rho} = 0.9996$), indicating that the score is not dominated by any single metric family. Sigmoid variation is effectively rank-invariant ($\rho \approx 1.000$ for all tested settings), confirming that this stage mostly rescales values without changing comparative ordering. The near-perfect

rank stability indicates that the aggregation structure dominates over parameter tuning, suggesting that the metric captures intrinsic structural signals rather than parameter-specific artifacts. Uniform weighting also remains highly consistent with the original formulation ($\rho = 0.9968$), suggesting that domain-informed weights refine but do not fundamentally determine rankings. In contrast, normalization choice is the main source of sensitivity (Min–Max / Z-score: $\rho_{\min} = 0.8557$, $\bar{\rho} = 0.8564$), implying that cross-ontology rank positions are more affected by dataset-dependent scaling than by weighting or sigmoid parameterization. This behavior is expected because Min–Max and Z-score are dependent on collections of ontologies, whereas $\log(1 + x)$ is pointwise and less dependent on the ontology collection. Overall, the complexity score is strongly robust to parametric and weighting choices, with normalization strategy being the primary methodological factor requiring explicit justification.

4.2.4 OntoLearner Software Access

OntoLearner is integrated with open-source platforms such as GitHub, HuggingFace, and PyPI, and is accompanied by comprehensive documentation on Read the Docs. These integrations promote transparency, reproducibility, and reuse of OL workflows across communities and domains.

GitHub and PyPI. OntoLearner’s source code is hosted on GitHub under the MIT License at <https://github.com/sciknoworg/OntoLearner>, with detailed contribution guidelines available at <https://github.com/sciknoworg/OntoLearner/blob/main/CONTRIBUTING.md>. The project is also published on PyPI for convenient distribution. An automated CI/CD pipeline supports streamlined releases across major operating systems (Windows, macOS, Linux) and multiple Python versions (N6).

Documentation. OntoLearner features extensive, well-structured documentation hosted on Read the Docs: <https://ontolearner.readthedocs.io/>. Key sections include: *Getting Started* – installation and quickstart guide; *Ontologizer* – instructions for ontology modularization, hosting, and adding new ontologies; *Learning Tasks* – supported OL tasks and usage of benchmarked ontologies; *Learning Models* – guidance for using LLMs in learning tasks; *Benchmarking* – provides comprehensive documentation for **180** modularized ontologies across **22** domains are currently added to OntoLearner (F4); *Package References* – detailed API reference. Documentation is automatically updated through a CI/CD pipeline, ensuring it stays current without interfering with benchmarking or library releases.

HuggingFace. OntoLearner’s end-to-end benchmarking is supported via HuggingFace, leveraged as a centralized hub for storing ontologies and learning task datasets across domains. Repositories are publicly accessible at <https://huggingface.co/SciKnowOrg> under the MIT License (F3, F4, N1). The ontologies and datasets are typically stored in the widely used formats such as OWL, RDF, XML, or TTL and are fully compatible with OntoLearner’s processing workflows. Moreover, the OL tasks datasets are stored in JSON. Users can upload their own ontologies to HuggingFace either through their personal account or by contributing directly to the SciKnowOrg organization page. Users can contribute by following the OntoLearner guidelines via https://ontolearner.readthedocs.io/ontologizer/new_ontologies.html, or by submitting a pull request with ontology files and metadata to the SciKnowOrg GitHub repository for review and integration.

4.3 Architectural Design and Compliance

4.3.1 Scalability and FAIR Alignment

Scalability. OntoLearner scales efficiently across large, multi-domain ontology collections. It currently integrates 180 ontologies across 22 domains, with an average of 3,401 classes per ontology—totaling 612,223 classes. The full pipeline, including parsing, importing, and dataset extraction for LLMs4OL tasks, completes in approximately 25.6 seconds per ontology, or 4.5 seconds overall. The Ontologizer component is designed to handle ontologies of all sizes—from small ontologies such as the Conference ontology [84] to large-scale ontologies such as ChEBI [68] with tens of thousands of classes—ensuring consistent performance regardless of scale.

Adherence to FAIR Principles. Ontologies in OntoLearner are richly annotated (e.g., domain, version, license, and provenance) and discoverable via the documentation and Ontologizer for streamlined navigation, ensuring that they are both human- and machine-discoverable (*Findable*). They are openly hosted on widely used platforms such as HuggingFace and GitHub, enabling easy retrieval

and verification (*Accessible*). OntoLearner supports multiple ontology serialization standards (i.e., OWL, RDF, XML, and TTL), adopts community vocabularies, and provides programmatic usage for integration into third-party workflows, thereby promoting semantic compatibility across tools and domains (*Interoperable*). Finally, modular packaging, inheritance-based ontology definitions, and Pythonic import mechanisms ensure seamless reuse in downstream applications. Clear licensing and documentation further encourage community contributions, solidifying long-term sustainability of ontologies within the ecosystem (*Reusable*).

OntoLearner Dublin Core Metadata. The `OntoLearnerMetadataExporter`¹⁰ component in OntoLearner is designed to automatically generate Dublin Core (DCMI) [116] metadata for each ontology in the benchmarking suite. This adds another layer of compliance with FAIR principles. Moreover, this module outputs DCMI, Ontologizers metadata info in RDF/XML formats containing key fields—title, description, creator, license, format, version, last-updated date, domain, category, and download URL—compatible with Protégé, RDFLib, or Apache Jena. Metadata is organized under a top-level collection resource describing the entire OntoLearner benchmark, and generation is automated via CI/CD workflows triggered on each new PyPI release.

4.3.2 Modular and Extensible Architecture

Internal Component Interactions. OntoLearner’s high-level view is represented in Figure 6, highlighting flexibility and scalability, enabling seamless integration of diverse ontologies through its modular and isolated architecture. This supports robust benchmarking and broad adaptability across ontological domains (F4) and end-to-end workflows. Moreover, the Figure 7a shows internal components and their interactions. Ontologies are first ingested and parsed by the `Ontologizer`, which extracts structural and semantic metadata for further processing. The `Text2Onto` module can then generate synthetic training data from the parsed ontologies, supporting term/type extraction tasks of OL. This data, along with existing datasets, feeds into the `LearnerPipeline` for automated model configuration and learning. The `Analyzer` computes ontological metrics, while the `Processor` coordinates outputs: testing, documentation updates, and publishing to HuggingFace. Later, OntoLearner metadata will be exported with a collection of Ontologizers. Finally, all `Metrics` are consolidated to maintain an up-to-date benchmark space¹¹, ensuring reproducibility and transparency across the library.

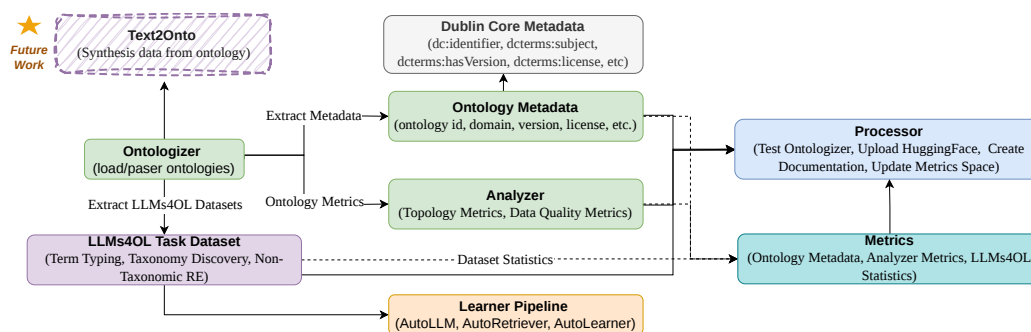
Modular Design for Ontologizer Extensibility. Ontologizer follows a modular design that simplifies ontology integration and supports extensibility with minimal overhead. As shown in Figure 7b, a generic `BaseOntology` class defines a reusable metadata schema covering core attributes. Domain-specific ontologies, such as ChEBI, inherit from this base class and populate the required fields, enabling rapid and consistent onboarding of new ontologies¹². This inheritance-based approach ensures that each ontology module remains self-contained, locally testable for metadata consistency, and easily integrable. Users can independently define, validate, and contribute modules via pull requests without altering core logic. This reduces duplication and complexity, enabling scalable, developer-friendly workflows aligned with sustainable software engineering principles. Moreover, the `train_test_split` used in Figure 7c uses Ontologizer to apply a leakage-aware split partitioning strategy that ensures the evaluation results for all three OL tasks are not inflated by hidden structural overlaps between splits. This shows how Ontologizer’s modular design is compatible with other components of OntoLearner.

Software Principles Adaptations. OntoLearner follows core software engineering principles (i.e., modularity, reusability, and separation of concerns) to provide a clear and extensible RAG framework. As illustrated in Figure 7c, lines 3–5 handle ontology loading and data extraction, line 6 performs data splitting, line 8 configures the LLM and prompting, line 9 initializes the retriever, and lines 10–15 perform learner initialization, training, and evaluation. Each of these steps is implemented as an independent component adhering to the Single Responsibility Principle (SRP) [71]. Retrievers (line 9) and LLMs (line 8) are pluggable through dependency injection, aligning with the Open/Closed Principle (OCP) [71], while learners depend only on the interfaces they require, following the Interface Segregation Principle (ISP) [51]. Loose coupling [70] is achieved by keeping model and retrieval logic independent of learning strategies, enabling new techniques to be added

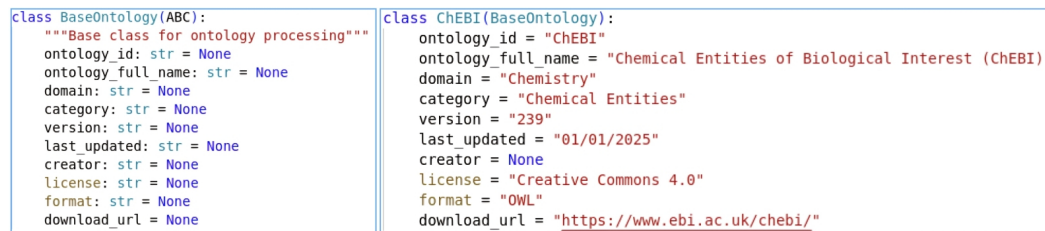
¹⁰Metadata documentation: <https://ontolearner.readthedocs.io/ontologizer/metadata.html>

¹¹Metric Space: <https://huggingface.co/spaces/SciKnowOrg/OntoLearner-Benchmark-Metrics>

¹²<https://github.com/sciknoworg/OntoLearner/blob/main/ontolearner/ontology/chemistry.py#L39>



(a) Architecture and module schema of the OntoLearner Python library. Planned future extensions are marked with a star.



(b) Illustrative example of ontology modularization within the Ontologizer component, showing a base ontology class alongside a modularized ChEBI ontology.

```

1 from ontolearner import AutoLLMLearner, AutoRetrieverLearner, AutoRAGLearner, MatOnto,
2   LabelMapper, StandardizedPrompting, train_test_split, evaluation_report
3
4 ontology = MatOnto()
5 ontology.load()
6 ontological_data = ontology.extract()
7 train_data, test_data = train_test_split(ontological_data, test_size=0.2)
8 task = 'term-typing'
9 llm = AutoLLMLearner(prompting=StandardizedPrompting, label_mapper=LabelMapper())
10 retriever = AutoRetrieverLearner(top_k=5)
11 rag = AutoRAGLearner(llm=llm, retriever=retriever)
12 rag.load(retriever_id='all-MiniLM-L6-v2', llm_id='Qwen/Qwen2.5-0.5B-Instruct')
13 rag.fit(train_data, task=task)
14 predicts = rag.predict(test_data, task=task)
15 truth = rag.tasks_ground_truth_former(data=test_data, task=task)
16 metrics = evaluation_report(y_true=truth, y_pred=predicts, task=task)

```

(c) Step-by-step implementation of a RAG-based learner in OntoLearner for the Term Typing task from LLMs4OL. The same workflow applies to other LLMs4OL tasks by modifying the task parameter.

```

1 from ontolearner import LearnerPipeline, MatOnto, train_test_split
2 ontology = MatOnto()
3 ontology.load()
4 ontological_data = ontology.extract()
5 train_data, test_data = train_test_split(ontological_data, test_size=0.2)
6 pipeline = LearnerPipeline(retriever_id='all-MiniLM-L6-v2', llm_id='Qwen/Qwen2.5-0.5B-Instruct',
7   hf_token='...', batch_size=32, top_k=5)
8 outputs = pipeline(train_data=train_data, test_data=test_data, evaluate=True, task='term-typing')
9 print("Metrics:", outputs['metrics'])
10 print("Elapsed time:", outputs['elapsed_time'])

```

(d) Example usage of the unified LearnerPipeline interface in OntoLearner for the Term Typing task, providing an end-to-end training and evaluation workflow.

Figure 7: OntoLearner Architectural Design.

without refactoring. As shown in Figure 7d, the LearnerPipeline (lines 6–8) further abstracts this workflow into a unified interface, reducing boilerplate code, promoting DRY (Don't Repeat Yourself) practices, and enabling reproducible experimentation across all LLM4OL tasks, similar to modern ML frameworks such as Transformers [118].

Table 5: Benchmark ontologies for experimental analysis with their documentation page at ontolearner.readthedocs.io.

ID	Ontology Full Name (Link)	Domain
DoCO	Document Components Ontology [21] (doco)	Education
Conference	Conference Ontology [84] (conference)	Events
GoodRelations	Good Relations Language Reference [48] (goodrelations)	Finance
Wine	Wine Ontology [41] (wine)	Food & Beverage
AgrO	Agronomy Ontology [4] (agro)	Agriculture
FoodOn	Food Ontology [29] (foodon)	
GeoNames	GeoNames Ontology [1] (geonames)	Geography
MaterialInformation	Material Information Ontology [3] (materialinformation)	Materials Science & Engineering
MatOnto	Material Ontology [18] (matonto)	
PeriodicTable	Periodic Table of the Elements Ontology [22] (periodictable)	
MDSOnto	Materials Data Science Ontology [91] (mdsonto)	
GO	Gene Ontology [2, 20] (go)	Biology & Life Sciences
ENVO	Environment Ontology [14, 15] (envo)	Ecology & Environment
SWEET	Semantic Web for Earth and Environment Technology Ontology [92] (sweet)	
CCO	Common Core Ontologies [52] (cco)	General Knowledge
DBpedia	DBpedia Ontology [5] (dbpedia)	
SchemaOrg	Schema.org Ontology [100] (schemaorg)	
AUTO	Automotive Ontology [33] (auto)	Industry
PTO	Product Types Ontology [49] (pto)	
DOID	Human Disease Ontology [102] (doid)	Medicine
OBi	Ontology for Biomedical Investigations [9] (obi)	
OM	Ontology of Units of Measure [59] (om)	Units and Measurements
QUDT	Quantities, Units, Dimensions and Data Types [31] (qudt)	
PROCO	PROcess Chemistry Ontology [99, 45] (proco)	Chemistry
VIBSO	Vibrational Spectroscopy Ontology [106] (vibso)	
ChEBI	CHEBI Integrated Role Ontology [68] (chebi)	

Learner Adaptations. OntoLearner, in addition to its core architecture, integrates learners from teams participating in the LLM4OL challenge¹³(F12). In particular, learners from the top-performing teams *RWTH-DBIS* [88], *SKH-NLP* [44], *Alexbek* [10], and *SBU-NLP* [90] have been incorporated to support benchmarking of diverse approaches. The *RWTH-DBIS* learner applies knowledge-enhanced continual learning with prompt-tuned LLMs for term typing and taxonomy discovery. The *SKH-NLP* learner combines BERT and LLaMA-based models for taxonomy discovery. The *Alexbek* learner uses heterogeneous methods, including few-shot prompting, ensemble typing, and attention-based graph modeling, to address the full OL pipeline. The *SBU-NLP* learner follows a training-free strategy using prompt engineering for stage-wise ontology construction. Further details on these learners are available in the OntoLearner documentation website¹⁴.

4.4 Experimental Setup

4.4.1 Selected Benchmark Ontologies

To ensure a comprehensive and unbiased evaluation, we selected a **26** diverse set of benchmark ontologies spanning multiple domains as summarized in Table 5. The selection was guided by three main criteria: domain diversity, structural and semantic heterogeneity, and community relevance. The benchmark set covers a broad spectrum of application domains, including education, events, finance, food and agriculture, geography, materials science and engineering, biology and life sciences, ecology and environment, medicine, chemistry, industry, and general knowledge. This diversity allows us to assess the robustness and generalizability of the evaluated methods across fundamentally different conceptual domains, ranging from highly specialized scientific ontologies (e.g., Gene Ontology [2], ChEBI [68], OBI [9]) to cross-domain and lightweight vocabularies (e.g., Schema.org). Moreover, the selected ontologies exhibit substantial variation in size, modeling granularity, and logical complexity. Some ontologies, such as GeoNames [1] and PeriodicTable [22], primarily provide taxonomic or descriptive structures with limited axiomatization, while others, such as GoodRelations [48], AgrO [4], FoodOn [29], MDSOnto [91], ChEBI, and the Common Core [52] ontologies, make extensive use of formal constraints, rich class hierarchies, and object property semantics. This heterogeneity is essential for evaluating how well the approaches scale and adapt to different ontology engineering practices and levels of expressivity.

For domains where multiple complementary ontologies exist (e.g., agriculture, materials science, ecology, medicine, chemistry, and units of measurement), we deliberately included more than one

¹³Challenge Website: <https://sites.google.com/view/llms4ol>

¹⁴Learners at OntoLearner: <https://ontolearner.readthedocs.io/learners/llms4ol.html>

ontology to capture intra-domain variability. This enables a finer analysis of domain-specific effects and reduces the risk of concluding isolated modeling choices. Overall, the benchmark comprises 26 of the 180 ontologies from the OntoLearner collection, yielding a balanced and representative testbed for cross-domain experimental analysis.

4.4.2 Selected Retrievers and Large Language Models

Retrievers. The $f(\cdot)$ is an encoder for representing a query Q and knowledge base (terms, types, and relationships) in a high-dimensional space. OntoLearner supports a diverse set of retrievers spanning lexical, embedding-based, and hybrid architecture retrieval models to enable fine-grained selections depending on the task at hand.

- *Lexical Retrievers.* We include several lexical baselines based on n-gram matching. These consist of word-level unigram frequency counting (Ngram-W-Counter) and word-level TF-IDF with sublinear term-frequency scaling $(1 + \log(tf))$ (Ngram-W-TFIDF). In addition, we employ character-level 3-gram frequency counting (Ngram-C-Counter) and character-level 3-gram TF-IDF with sublinear scaling (Ngram-C-TFIDF). To capture mixed lexical patterns, we further include character–word 3-gram frequency counting (Ngram-CW-Counter) and character–word 3-gram TF-IDF (Ngram-CW-TFIDF).
- *Traditional Word Embeddings.* We evaluate classical static embedding models, including GloVe [89] using the Stanford GloVe Dolma model trained on 220B tokens with a 1.2M vocabulary and 300-dimensional uncased vectors, as well as Word2Vec [72] using the Google Word2Vec embeddings with 300-dimensional vectors.
- *Sentence Transformers.* We include multiple sentence-level bi-encoder models from the Sentence Transformers framework [95]. MiniLM-L6 is a lightweight general-purpose bi-encoder trained on over one billion sentence pairs to produce normalized 384-dimensional embeddings (all-MiniLM-L6-v2). MPNET is a higher-capacity bi-encoder based on the MPNet architecture with mean pooling, trained on over one billion text pairs to generate normalized 768-dimensional embeddings (all-mpnet-base-v2). GTR-T5 [75] is a dual-encoder architecture trained specifically for semantic search, producing 768-dimensional dense representations (gtr-t5-base). We also include domain-specific models, namely MatSciBERT [43], a BERT model trained on materials science literature for improved information extraction (matscibert), and BiomedBERT [42], a BERT-based model pretrained from scratch on PubMed abstracts that achieves state-of-the-art performance on biomedical NLP benchmarks (BiomedNLP-BiomedBERT-base-uncased-abstract).
- *Modern Embedding Models (Bi-Encoders).* We further evaluate modern large-scale embedding models. Nomic [83] is an open-source text embedding model with long-context support that outperforms OpenAI Ada-002 on both short- and long-context benchmarks (nomic-embed-text-v1.5). Nomic-MoE [81] extends this approach with a Mixture-of-Experts architecture, improving efficiency while maintaining competitive performance (nomic-embed-text-v2-moe). We also include Qwen3 Embeddings [124], a family of proprietary multilingual embedding models available in 0.6B, 4B, and 8B parameter sizes (Qwen3-0.6B, Qwen3-4B, and Qwen3-8B). These models support long-context understanding, flexible embedding dimensions, and achieve state-of-the-art performance on benchmarks such as MTEB [73]. Finally, we include Gemma [115], a lightweight multilingual embedding model optimized for on-device inference, supporting over 100 languages with low latency and efficient memory usage (embeddinggemma-300m).
- *Hybrid Cross-Encoder and Bi-Encoder Retrieval.* To balance efficiency and accuracy, we include a cascaded retrieval system, CS-Qwen3-8B-MiniLM [95]. This hybrid approach combines a Qwen3-Embedding-8B bi-encoder for efficient candidate retrieval with a MiniLM-L12 cross-encoder fine-tuned on MS MARCO for accurate reranking, enabling scalable yet precise retrieval.

Large Language Models. We conducted experiments using a diverse collection of LLMs spanning multiple model families, parameter scales, and training paradigms, comprising a total of 12 models. From the Qwen family [120], we evaluated base, instruction-tuned, and reasoning-oriented variants, including Qwen3-0.6B, Qwen3-4B-Instruct-2507, Qwen3-4B-Thinking-2507, Qwen3-8B, Qwen3-14B, and the large-scale mixture-of-experts model Qwen3-Next-80B-A3B-Instruct. Similarly, from the Gemma family [57], we included instruction-tuned models at varying scales, namely Gemma-3-1b-it, Gemma-3-4b-it, Gemma-3-12b-it, and Gemma-3-27b-it. In addition, we evaluated

the instruction-tuned Mistral-Small-3.2-24B-Instruct-2506 model [53], as well as a model from the Falcon family [125], specifically Falcon-H1-1.5B-Deep-Instruct. This comprehensive selection enables an in-depth analysis of the effects of model family, scale, and training strategy (e.g., instruction tuning and explicit reasoning objectives).

4.4.3 Retrieval-Augmented Generation Pipeline

The RAG module consists of two main components: a retriever and an LLM-based re-ranker. Let Q denote an input query, the retriever defines a scoring function $s : Q \times \mathcal{C} \rightarrow \mathbb{R}$, where $\mathcal{C} = \{c_1, c_2, \dots, c_N\}$ is the set of candidates. It selects the top- k most relevant candidates as $\mathcal{C}_k(Q) = \text{TopK}_{c \in \mathcal{C}} s(Q, c)$. Given $\mathcal{C}_k(Q)$, the LLM estimates the relevance of each candidate via $LLM_r : Q \times \mathcal{C} \rightarrow [0, 1]$, and the final selection is $\mathcal{C}^*(Q) = \{c \in \mathcal{C}_k(Q) \mid LLM_r(Q, c) = 1\}$. The scoring function uses cosine similarity, $s(Q, c) = \cos(q, c_d) = \frac{q^T c_d}{\|q\|_2 \|c_d\|_2}$, where the query Q and a candidate document $d \in \mathcal{D}$ are mapped to a shared embedding space using an encoder $q = f(Q)$ and $c_d = f(d)$. Retrieval performance, therefore, depends on the geometric alignment between query and candidate embeddings¹⁵.

Term Typing. This task is defined as: *Given a lexical term, identify its types*. Let a lexical term $\mathcal{L}$ be the query and $\mathcal{T}$ the set of candidate types. The retriever selects $\mathcal{T}_k(\mathcal{L}) = \text{TopK}_{t \in \mathcal{T}} s(\mathcal{L}, t)$, where $\mathcal{T} = \{t_1, t_2, \dots, t_N\}$. Since a term may belong to multiple types, the LLM selects $\mathcal{T}^*(\mathcal{L}) = \{t \in \mathcal{T}_k(\mathcal{L}) \mid LLM_r(\mathcal{L}, t) = 1\}$.

Type Taxonomy Discovery. This task involves “*Given the term types, discover the taxonomy hierarchy between them*”. It focuses on constructing hierarchical relationships between types (i.e., is-a relationships). The OntoLearner supports two types of retrievers, described as follows:

- **Traditional Retriever.** Consider a taxonomy $\mathcal{T}$ represented as a directed acyclic graph $\mathcal{H} = (\mathcal{T}, \epsilon)$, where $(t_i, t_j) \in \epsilon$ indicates $t_i \sqsubseteq t_j$. In the RAG pipeline, taxonomy discovery is formulated as pairwise hierarchical relation prediction. Given $t_q \in \mathcal{T}$, the retriever scores all types as potential parent candidates using $s : \mathcal{T} \times \mathcal{T} \rightarrow \mathbb{R}$ and selects the top- k candidates $\mathcal{T}_k(t_q) = \text{TopK}_{t \in \mathcal{T}} s(t_q, t)$. The LLM then acts as a hierarchical verifier, $LLM_h : \mathcal{T} \times \mathcal{T} \rightarrow [0, 1]$, where $LLM_h(t_q, t_p) = 1$ indicates that t_q is a subclass of t_p . The final predicted parent types are $\mathcal{P}^*(t_q) = \{t_p \in \mathcal{T}_k(t_q) \mid LLM_h(t_q, t_p) = 1\}$. Aggregating across all query types yields the discovered taxonomy $\epsilon^* = \bigcup_{t_q \in \mathcal{T}} \{(t_q, t_p) \mid t_p \in \mathcal{P}^*(t_q)\}$, $\mathcal{H}^* = (\mathcal{T}, \epsilon^*)$. Following the OntoLearner architecture, this task reuses the term typing formulation, with $t_q \in \mathcal{T}$ as the query instead of $\mathcal{L}$.
- **LLM-Augmented Retriever.** This approach extends traditional retrievers by introducing an LLM-driven candidate generation stage prior to retrieval. Given $t_q \in \mathcal{T}$, the LLM first infers a coarse set of plausible parent candidates using generative AI. Formally, the LLM defines a candidate proposal function $LLM_c : \mathcal{T} \rightarrow \mathcal{T}_n$, producing a top- n set $\mathcal{T}_n^{LLM}(t_q) = LLM_c(t_q)$, where $|\mathcal{T}_n^{LLM}(t_q)| = n$, and each $t_p \in \mathcal{T}_n^{LLM}(t_q)$ is a potential (t_q, t_p) is-a relation. The retriever then operates on $\mathcal{T}_n^{LLM}(t_q)$ rather than the full type set. Using the scoring function $s : \mathcal{T} \times \mathcal{T} \rightarrow \mathbb{R}$, the top- k candidates are $\mathcal{T}_k(t_q) = \text{TopK}_{t \in \mathcal{T}_n^{LLM}(t_q)} s(t_q, t)$, and the hierarchical verifier $LLM_h : \mathcal{T} \times \mathcal{T} \rightarrow [0, 1]$ determines valid subclass relations using $\mathcal{P}^*(t_q) = \{t_p \in \mathcal{T}_k(t_q) \mid LLM_h(t_q, t_p) = 1\}$. By constraining the candidate space, this approach improves semantic alignment while preserving the modularity of the RAG-based taxonomy discovery pipeline.

Non-Taxonomic Relation Extraction. This task is defined as: *Given two types, identify if there is a non-taxonomic semantic relationship between them, and if so, which one*. Unlike taxonomy discovery, which focuses on hierarchical is-a relations, this task captures other meaningful associations (e.g., part-of, used-for, causes, related-to) to enrich the ontology. Let $\mathcal{R}$ denote the set of predefined non-taxonomic relation types. The task is formulated as a multi-stage RAG-based process. First, candidate type pairs are identified using the taxonomy discovery formulation: for $t_q \in \mathcal{T}$, the retriever selects $\mathcal{T}_k(t_q)$, yielding type pairs (t_q, t_c) likely to exhibit a non-taxonomic relation. Second, each pair (t_q, t_c) is converted into a natural language query $Q_{(t_q, t_c)}$ using a verbalization template,

¹⁵A single executable pipeline that instantiates all configurations described in this section is provided in the OntoLearner repository <https://github.com/sciknoworg/OntoLearner/blob/main/examples/pipeline.ipynb> and through the documentation website.

887 and the retriever selects top- k relation candidates: $\mathcal{R}_k(t_q, t_c) = \text{TopK}_{r \in \mathcal{R}} s(Q(t_q, t_c), r)$, where $s(\cdot)$
888 is the RAG similarity function. Finally, the LLM acts as a relation verifier with prediction function
889 $LLM_r : \mathcal{T} \times \mathcal{T} \times \mathcal{R} \rightarrow [0, 1]$, where $LLM_r(t_q, t_c, r) = 1$ indicates that relation r holds. The final
890 predicted relations are $\mathcal{R}^*(t_q, t_c) = \{r \in \mathcal{R}_k(t_q, t_c) \mid LLM_r(t_q, t_c, r) = 1\}$. Aggregating over all
891 queries gives the discovered set:

$$\mathcal{E}_R^* = \bigcup_{t_q \in \mathcal{T}} \bigcup_{t_c \in \mathcal{T}_k(t_q)} \{(t_q, r, t_c) \mid r \in \mathcal{R}^*(t_q, t_c)\}.$$

892 Following the OntoLearner design, this task reuses the taxonomy discovery and term typing formu-
893 lations with minimal modifications.

894 Data availability

895 The OntoLearner benchmark collections are available at [https://huggingface.co/collections/SciKnowOrg/ontolearner-](https://huggingface.co/collections/SciKnowOrg/ontolearner-benchmarking)
896 benchmarking.

897 Code availability

898 Source code for OntoLearner is available under the MIT License at [https://github.com/](https://github.com/sciknoworg/OntoLearner/)
899 [sciknoworg/OntoLearner/](https://github.com/sciknoworg/OntoLearner/), with documentation at [https://ontolearner.readthedocs.](https://ontolearner.readthedocs.io/)
900 [io/](https://ontolearner.readthedocs.io/).

901 Acknowledgements

902 This work is jointly supported by the SCINEXT project (BMFTR, German Federal Ministry of Re-
903 search, Technology and Space, Grant ID: 01IS22070), the KISSKI AI Service Center (BMFTR,
904 Grant ID: 01IS22093C), the NFDI4DataScience initiative (DFG, German Research Foundation,
905 Grant ID: 460234259), and the NFDI4ING initiatives (DFG, German Research Foundation, Grant
906 ID: 442146713).

907 References

- 908 [1] Geonames ontology (geonames). <https://www.geonames.org/ontology>, 2022. RDF
909 ontology. Licensed under Creative Commons 3.0. Last updated: 2022-01-30.
- 910 [2] Michael Ashburner, Catherine A Ball, Judith A Blake, David Botstein, Heather Butler,
911 J Michael Cherry, Allan P Davis, Kara Dolinski, Selina S Dwight, Janan T Eppig, et al.
912 Gene ontology: tool for the unification of biology. *Nature genetics*, 25(1):25–29, 2000.
- 913 [3] Toshihiro Ashino. Materials ontology: An infrastructure for exchanging materials informa-
914 tion and knowledge. *Data Science Journal*, 9:54–61, 2010.
- 915 [4] Céline Aubert, Marie-Angélique Laporte, Krishna Udaiwal, Pier Luigi Buttigieg, Chris
916 Mungall, Elizabeth Arnaud, and Charles Tapley Hoyt. *Agriculturalsemantics/agro*: Novem-
917 ber 2022 release, November 2022.
- 918 [5] Sören Auer, Christian Bizer, Georgi Kobilarov, Jens Lehmann, Richard Cyganiak, and
919 Zachary Ives. Dbpedia: A nucleus for a web of open data. In *international semantic web*
920 *conference*, pages 722–735. Springer, 2007.
- 921 [6] Hamed Babaei Giglou, Jennifer D’Souza, and Sören Auer. *LLMs4Synthesis: Leveraging*
922 *Large Language Models for Scientific Synthesis*. Association for Computing Machinery, New
923 York, NY, USA, 2025.
- 924 [7] Hamed Babaei Giglou, Jennifer D’Souza, Nandana Mihindukulasooriya, and Sören Auer.
925 Llm4ol 2025 overview: The 2nd large language models for ontology learning challenge.
926 *Open Conference Proceedings*, 2025.

- 927 [8] Hamed Babaei Giglou, Jennifer D’Souza, and Sören Auer. Llms4ol: Large language models
928 for ontology learning. In *International Semantic Web Conference*, pages 408–427. Springer,
929 2023.
- 930 [9] Anita Bandrowski, Ryan Brinkman, Mathias Brochhausen, Matthew H Brush, Bill Bug, Mar-
931 cus C Chibucos, Kevin Clancy, Mélanie Courtot, Dirk Derom, Michel Dumontier, et al. The
932 ontology for biomedical investigations. *PloS one*, 11(4):e0154556, 2016.
- 933 [10] Aleksandra Beliaeva and Temurbek Rahmatullaev. Alexbek at llms4ol 2025 tasks a, b, and
934 c: Heterogeneous llm methods for ontology learning (few-shot prompting, ensemble typing,
935 and attention-based taxonomies). In *Open Conference Proceedings*, volume 6, 2025.
- 936 [11] Bikram Pratim Bhuyan, Amar Ramdane-Cherif, Ravi Tomar, and TP Singh. Neuro-symbolic
937 artificial intelligence: a survey. *Neural Computing and Applications*, 36(21):12809–12844,
938 2024.
- 939 [12] BioPortal. FAIRsharing.org: BioPortal. [https://doi.org/10.25504/FAIRsharing.](https://doi.org/10.25504/FAIRsharing.4m97ah)
940 [4m97ah](https://doi.org/10.25504/FAIRsharing.4m97ah), March 2024. Last accessed: 2025-05-02.
- 941 [13] Olivier Bodenreider. The unified medical language system (umls): integrating biomedical
942 terminology. *Nucleic acids research*, 32(suppl_1):D267–D270, 2004.
- 943 [14] Pier Luigi Buttigieg, Norman Morrison, Barry Smith, Christopher J Mungall, Suzanna E
944 Lewis, and Envo Consortium. The environment ontology: contextualising biological and
945 biomedical entities. *Journal of biomedical semantics*, 4(1):43, 2013.
- 946 [15] Pier Luigi Buttigieg, Evangelos Pafilis, Suzanna E Lewis, Mark P Schildhauer, Ramona L
947 Walls, and Christopher J Mungall. The environment ontology in 2016: bridging domains
948 with increased scope, semantic density, and interoperability. *Journal of biomedical semantics*,
949 7(1):57, 2016.
- 950 [16] Andrew Carlson, Justin Betteridge, Bryan Kisiel, Burr Settles, Estevam Hruschka, and Tom
951 Mitchell. Toward an architecture for never-ending language learning. In *Proceedings of the*
952 *AAAI conference on artificial intelligence*, volume 24, pages 1306–1313, 2010.
- 953 [17] J Harry Caufield, Harshad Hegde, Vincent Emonet, Nomi L Harris, Marcin P Joachimiak,
954 Nicolas Matentzoglou, HyeonSik Kim, Sierra Moxon, Justin T Reese, Melissa A Haen-
955 del, et al. Structured prompt interrogation and recursive extraction of semantics (spires):
956 A method for populating knowledge bases using zero-shot learning. *Bioinformatics*,
957 40(3):btac104, 2024.
- 958 [18] Kwok Cheung, John Drennan, and Jane Hunter. Towards an ontology for data-driven discov-
959 ery of new materials. In *AAAI Spring Symposium: Semantic Scientific Knowledge Integration*,
960 pages 9–14, 2008.
- 961 [19] Philipp Cimiano and Johanna Völker. Text2onto: A framework for ontology learning and
962 data-driven change discovery. In *International conference on application of natural language*
963 *to information systems*, pages 227–238. Springer, 2005.
- 964 [20] The Gene Ontology Consortium, Suzi A Aleksander, James Balhoff, Seth Carbon, J Michael
965 Cherry, Harold J Drabkin, Dustin Ebert, Marc Feuermann, Pascale Gaudet, Nomi L Harris,
966 David P Hill, Raymond Lee, Huaiyu Mi, Sierra Moxon, Christopher J Mungall, Anushya
967 Muruganugan, Tremayne Mushayahama, Paul W Sternberg, Paul D Thomas, Kimberly
968 Van Auker, Jolene Ramsey, Deborah A Siegle, Rex L Chisholm, Petra Fey, Maria Cristina
969 Aspromonte, Maria Victoria Nugnes, Federica Quaglia, Silvio Tosatto, Michelle Giglio, Su-
970 varna Nadendla, Giulia Antonazzo, Helen Attrill, Gil dos Santos, Steven Marygold, Victor
971 Strelets, Christopher J Tabone, Jim Thurmond, Pinglei Zhou, Saadullah H Ahmed, Praoparn
972 Asanithong, Diana Luna Buitrago, Meltem N Erdol, Matthew C Gage, Mohamed Ali Kad-
973 hum, Kan Yan Chloe Li, Miao Long, Aleksandra Michalak, Angeline Pesala, Armalya Pri-
974 tazahra, Shirin C C Saverimuttu, Renzhi Su, Kate E Thurlow, Ruth C Lovering, Colin
975 Logie, Snezhana Oliferenko, Judith Blake, Karen Christie, Lori Corbani, Mary E Dolan,
976 Harold J Drabkin, David P Hill, Li Ni, Dmitry Sitnikov, Cynthia Smith, Alayne Cuzick,
977 James Seager, Laurel Cooper, Justin Elser, Pankaj Jaiswal, Parul Gupta, Pankaj Jaiswal,

- 978 Sushma Naithani, Manuel Lera-Ramirez, Kim Rutherford, Valerie Wood, Jeffrey L De Pons,
979 Melinda R Dwinell, G Thomas Hayman, Mary L Kaldunski, Anne E Kwitek, Stanley J F
980 Laulederkind, Marek A Tutaj, Mahima VEDI, Shur-Jen Wang, Peter D'Eustachio, Lucila
981 Aimo, Kristian Axelsen, Alan Bridge, Nevila Hyka-Nouspikel, Anne Morgat, Suzi A Alek-
982 sander, J Michael Cherry, Stacia R Engel, Kalpana Karra, Stuart R Miyasato, Robert S Nash,
983 Marek S Skrzypek, Shuai Weng, Edith D Wong, Erika Bakker, Tanya Z Berardini, Leonore
984 Reiser, Andrea Auchincloss, Kristian Axelsen, Ghislaine Argoud-Puy, Marie-Claude Blatter,
985 Emmanuel Boutet, Lionel Breuza, Alan Bridge, Cristina Casals-Casas, Elisabeth Coudert,
986 Anne Estreicher, Maria Livia Famiglietti, Marc Feuermann, Arnaud Gos, Nadine Gruaz-
987 Gumowski, Chantal Hulo, Nevila Hyka-Nouspikel, Florence Jungo, Philippe Le Mercier,
988 Damien Lieberherr, Patrick Masson, Anne Morgat, Ivo Pedruzzi, Lucille Pourcel, Sylvain
989 Poux, Catherine Rivoire, Shyamala Sundaram, Alex Bateman, Emily Bowler-Barnett, Hema
990 Bye-A-Jee, Paul Denny, Alexandr Ignatchenko, Rizwan Ishtiaq, Antonia Lock, Yvonne Lussi,
991 Michele Magrane, Maria J Martin, Sandra Orchard, Pedro Raposo, Elena Speretta, Nidhi
992 Tyagi, Kate Warner, Rossana Zaru, Alexander D Diehl, Raymond Lee, Juancarlos Chan,
993 Stavros Diamantakis, Daniela Raciti, Magdalena Zarowiecki, Malcolm Fisher, Christina
994 James-Zorn, Virgilio Ponferrada, Aaron Zorn, Sridhar Ramachandran, Leyla Ruzicka, and
995 Monte Westerfield. The gene ontology knowledgebase in 2023. *Genetics*, 224(1):iyad031,
996 03 2023.
- 997 [21] Alexandru Constantin, Silvio Peroni, Steve Pettifer, David Shotton, and Fabio Vitali. The
998 document components ontology (doco). *Semantic web*, 7(2):167–181, 2016.
- 999 [22] Michael Cook. Periodic table of the elements ontology (periodictable). <https://www.daml.org/2003/01/periodictable/>, February 2004. OWL ontology. Last updated: 2004-02-
1000 05.
1001
- 1002 [23] Caglar Demir, Alkid Baci, N'Dah Jean Kouagou, Leonie Nora Sieger, Stefan Heindorf, Si-
1003 mon Bin, Lukas Blübaum, Alexander Biggerl, and Axel-Cyrille Ngonga Ngomo. Ontolearn—
1004 a framework for large-scale owl class expression learning in python. *Journal of Machine*
1005 *Learning Research*, 26(63):1–6, 2025.
- 1006 [24] Caglar Demir and Axel-Cyrille Ngonga Ngomo. Neuro-symbolic class expression learning.
1007 In *The 32nd International Joint Conference on Artificial Intelligence, IJCAI 2023*, 2023.
- 1008 [25] de.NBI. Ontologies and terminology services ecosystem in life sci-
1009 ences and beyond. [https://www.denbi.de/de-nbi-events-archive/](https://www.denbi.de/de-nbi-events-archive/1604-ontologies-and-terminology-services-ecosystem-in-life-sciences-and-beyond)
1010 1604-ontologies-and-terminology-services-ecosystem-in-life-sciences-and-beyond,
1011 n.d. Accessed: 2025-05-02.
- 1012 [26] Jacob Devlin, Ming-Wei Chang, Kenton Lee, and Kristina Toutanova. Bert: Pre-training
1013 of deep bidirectional transformers for language understanding. In *Proceedings of the 2019*
1014 *conference of the North American chapter of the association for computational linguistics:*
1015 *human language technologies, volume 1 (long and short papers)*, pages 4171–4186, 2019.
- 1016 [27] Diffbot. The knowledge graph of the public web. [https://www.diffbot.com/products/](https://www.diffbot.com/products/knowledge-graph/)
1017 [knowledge-graph/](https://www.diffbot.com/products/knowledge-graph/). Accessed: 2026-01-27. Description: Structured knowledge graph of
1018 over 10 billion public web entities with 50+ data fields for news, organizations, people, prod-
1019 ucts, and more.
- 1020 [28] Yihong Dong, Xue Jiang, Huanyu Liu, Zhi Jin, Bin Gu, Mengfei Yang, and Ge Li. General-
1021 ization or memorization: Data contamination and trustworthy evaluation for large language
1022 models. In *Findings of the Association for Computational Linguistics: ACL 2024*, pages
1023 12039–12050, 2024.
- 1024 [29] Damion M Dooley, Emma J Griffiths, Gurinder S Gosal, Pier L Buttigieg, Robert Hoehndorf,
1025 Matthew C Lange, Lynn M Schriml, Fiona SL Brinkman, and William WL Hsiao. Foodon:
1026 a harmonized food ontology to increase global food traceability, quality control and data
1027 integration. *npj Science of Food*, 2(1):23, 2018.
- 1028 [30] Markus Eberts and Adrian Ulges. Span-based joint entity and relation extraction with trans-
1029 former pre-training. *arXiv preprint arXiv:1909.07755*, 2019.

- 1030 [31] FAIRsharing.org. QUDT: Quantities, Units, Dimensions and Types. Online, 2025. Accessed:
1031 2026-01-06; Last edited: 2025-11-03.
- 1032 [32] Nadeen Fathallah, Arunav Das, Stefano De Giorgis, Andrea Poltronieri, Peter Haase, and Li-
1033 ubov Kovriguina. Neon-gpt: a large language model-powered pipeline for ontology learning.
1034 In *European Semantic Web Conference*, pages 36–50. Springer, 2024.
- 1035 [33] Michael Feld and Christian Müller. The automotive ontology: managing knowledge inside
1036 the vehicle and sharing it between cars. In *Proceedings of the 3rd International Conference*
1037 *on Automotive User Interfaces and Interactive Vehicular Applications*, pages 79–86, 2011.
- 1038 [34] Blaz Fortuna, Marko Grobelnik, and Dunja Mladenic. Ontogen: Semi-automatic ontology
1039 editor. In *Symposium on human interface and the management of information*, pages 309–
1040 318. Springer, 2007.
- 1041 [35] Tianyu Gao, Xingcheng Yao, and Danqi Chen. Simcse: Simple contrastive learning of sen-
1042 tence embeddings. *arXiv preprint arXiv:2104.08821*, 2021.
- 1043 [36] Yunfan Gao, Yun Xiong, Xinyu Gao, Kangxiang Jia, Jinliu Pan, Yuxi Bi, Yi Dai, Jiawei Sun,
1044 Qianyu Guo, Meng Wang, et al. Retrieval-augmented generation for large language models:
1045 A survey. *CoRR*, 2023.
- 1046 [37] Laurent Gatto. An r interface to the ontology lookup service. [https://www.](https://www.bioconductor.org/packages/devel/bioc/vignettes/rols/inst/doc/rols.html)
1047 [bioconductor.org/packages/devel/bioc/vignettes/rols/inst/doc/rols.html](https://www.bioconductor.org/packages/devel/bioc/vignettes/rols/inst/doc/rols.html),
1048 2025. Bioconductor vignette, accessed May 3, 2025.
- 1049 [38] Hamed Babaei Giglou, Jennifer D’Souza, and Sören Auer. Llms4ol 2024 overview: The 1st
1050 large language models for ontology learning challenge. *arXiv preprint arXiv:2409.10146*,
1051 2024.
- 1052 [39] Hamed Babaei Giglou, Jennifer D’Souza, Nandana Mihindukulasooriya, and Sören Auer.
1053 Llms4ol 2025 overview: The 2nd large language models for ontology learning challenge. In
1054 *Open Conference Proceedings*, volume 6, 2025.
- 1055 [40] Luis Gouveia and Pedro Martins. A hierarchy of hop-indexed models for the capacitated
1056 minimum spanning tree problem. *Networks: An International Journal*, 35(1):1–16, 2000.
- 1057 [41] João Graça, Márcio Mourao, Orlando Anunciação, Pedro Monteiro, H Sofia Pinto, and
1058 Virgílio Loureiro. Ontology building process: the wine domain. In *Proc. of the 5th Conf.*
1059 *of EFITA*, 2005.
- 1060 [42] Yu Gu, Robert Tinn, Hao Cheng, Michael Lucas, Naoto Usuyama, Xiaodong Liu, Tristan
1061 Naumann, Jianfeng Gao, and Hoifung Poon. Domain-specific language model pretraining
1062 for biomedical natural language processing. *ACM Transactions on Computing for Healthcare*
1063 (*HEALTH*), 3(1):1–23, 2021.
- 1064 [43] Tanishq Gupta, Mohd Zaki, NM Anoop Krishnan, and Mausam. Matscibert: A materials
1065 domain language model for text mining and information extraction. *npj Computational Ma-*
1066 *terials*, 8(1):102, 2022.
- 1067 [44] Seyed Mohammad Hossein Hashemi, Mostafa Karimi Manesh, and Mehrnoush Shamsfard.
1068 Skh-nlp at llms4ol 2024 task b: Taxonomy discovery in ontologies using bert and llama 3. In
1069 *Open Conference Proceedings*, volume 4, pages 103–111, 2024.
- 1070 [45] Oliver He and Wes Schafer. Extending the allotrope framework: An ontological representa-
1071 tion and analysis of process chemistry. 2020 Fall Allotrope Connect Virtual Conference (Oral
1072 presentation), September 2020. Virtual conference, September 30, 2020.
- 1073 [46] Marti A Hearst. Automatic acquisition of hyponyms from large text corpora. In *COLING*
1074 *1992 volume 2: The 14th international conference on computational linguistics*, 1992.
- 1075 [47] Stefan Heindorf, Lukas Blübaum, Nick Düsterhus, Till Werner, Varun Nandkumar Golani,
1076 Caglar Demir, and Axel-Cyrille Ngonga Ngomo. Evolearner: Learning description logics
1077 with evolutionary algorithms. In *Proceedings of the ACM Web Conference 2022*, pages 818–
1078 828, 2022.

- [48] Martin Hepp. Goodrelations: An ontology for describing products and services offers on the web. In *International conference on knowledge engineering and knowledge management*, pages 329–346. Springer, 2008.
- [49] Martin Hepp, Katharina Siorpaes, and Daniel Bachlechner. Harvesting wiki consensus: Using wikipedia entries as vocabulary for knowledge management. *IEEE Internet Computing*, 11(5):54–65, 2007.
- [50] Pascal Hitzler, Monireh Ebrahimi, Md Kamruzzaman Sarker, and Daria Stepanova. Neuro-symbolic ai and the semantic web, 2024.
- [51] ED Hondros and MP Seah. Segregation to interfaces. *International Metals Reviews*, 22(1):262–301, 1977.
- [52] Mark Jensen, Giacomo De Colle, Sean Kindya, Cameron More, Alexander P Cox, and John Beverley. The common core ontologies. *arXiv preprint arXiv:2404.17758*, 2024.
- [53] Albert Q. Jiang, Alexandre Sablayrolles, Arthur Mensch, Chris Bamford, Devendra Singh Chaplot, Diego de las Casas, Florian Bressand, Gianna Lengyel, Guillaume Lample, Lucile Saulnier, L  lio Renard Lavaud, Marie-Anne Lachaux, Pierre Stock, Teven Le Scao, Thibaut Lavril, Thomas Wang, Timoth  e Lacroix, and William El Sayed. Mistral 7b, 2023.
- [54] Clement Jonquet, Anne Toulet, Elizabeth Arnaud, Sophie Aubin, Esther Dzal   Yeumo, Vincent Emonet, John Graybeal, Marie-Ang  lique Laporte, Mark A Musen, Valeria Pesce, et al. Agroportal: A vocabulary and ontology repository for agronomy. *Computers and Electronics in Agriculture*, 144:126–143, 2018.
- [55] Martin Josifoski, Marija Sakota, Maxime Peyrard, and Robert West. Exploiting asymmetry for synthetic training data generation: SynthIE and the case of information extraction. In *Proceedings of the 2023 Conference on Empirical Methods in Natural Language Processing*, pages 1555–1574, Singapore, December 2023. Association for Computational Linguistics.
- [56] Simon Jupp, Tony Burdett, Catherine Leroy, and Helen E Parkinson. A new ontology lookup service at embl-ebi. *SWAT4LS*, 2:118–119, 2015.
- [57] Aishwarya Kamath, Johan Ferret, Shreya Pathak, Nino Vieillard, Ramona Merhej, Sarah Perrin, Tatiana Matejovicova, Alexandre Ram  , Morgane Riv  re, Louis Rouillard, et al. Gemma 3 technical report. *CoRR*, 2025.
- [58] Maulik R Kamdar, Tania Tudorache, and Mark A Musen. A systematic analysis of term reuse and term overlap across biomedical ontologies. *Semantic web*, 8(6):853–871, 2017.
- [59] Jan Martin Keil and Sirko Schindler. Comparison and evaluation of ontologies for units of measurement. *Semantic Web*, 10(1):33–51, 2018.
- [60] Vamsi Krishna Kommineni, Birgitta K  nig-Ries, and Sheeba Samuel. From human experts to machines: An llm supported approach to ontology and knowledge graph construction. *CoRR*, 2024.
- [61] Angelina Kraft, Felix Engel, and Oliver Koepler. Tib terminology service – terminologies and ontologies support fair research data management, 2023. The TIB Terminology Service provides access to terminologies from domains such as architecture, chemistry, computer science, mathematics, and physics. First accessed on 2023-09-07.
- [62] Tobias Kuhn. A survey and classification of controlled natural languages. *Computational linguistics*, 40(1):121–170, 2014.
- [63] Patrick Lewis, Ethan Perez, Aleksandra Piktus, Fabio Petroni, Vladimir Karpukhin, Naman Goyal, Heinrich K  ttler, Mike Lewis, Wen-tau Yih, Tim Rockt  schel, et al. Retrieval-augmented generation for knowledge-intensive nlp tasks. *Advances in neural information processing systems*, 33:9459–9474, 2020.
- [64] Aixin Liu, Bei Feng, Bing Xue, Bingxuan Wang, Bochao Wu, Chengda Lu, Chenggang Zhao, Chengqi Deng, Chenyu Zhang, Chong Ruan, et al. Deepseek-v3 technical report. *arXiv preprint arXiv:2412.19437*, 2024.

- 1128 [65] Andy Lo, Albert Q Jiang, Wenda Li, and Mateja Jamnik. End-to-end ontology learning
1129 with large language models. *Advances in Neural Information Processing Systems*, 37:87184–
1130 87225, 2024.
- 1131 [66] Anemily Machina and Robert Mercer. Anisotropy is not inherent to transformers. In Kevin
1132 Duh, Helena Gomez, and Steven Bethard, editors, *Proceedings of the 2024 Conference of the*
1133 *North American Chapter of the Association for Computational Linguistics: Human Language*
1134 *Technologies (Volume 1: Long Papers)*, pages 4892–4907, Mexico City, Mexico, June 2024.
1135 Association for Computational Linguistics.
- 1136 [67] Alexander Maedche. *Ontology Learning for the Semantic Web*, volume 665 of *The Springer*
1137 *International Series in Engineering and Computer Science*. Springer New York, 2002.
- 1138 [68] Adnan Malik, Muhammad Arsalan, Carlos Moreno, Juan Mosquera, Eloy Félix, Tev-
1139 fik Kizilören, Venkatesh Muthukrishnan, Barbara Zdrazil, Andrew R Leach, and Noel M
1140 O’Boyle. Chebi: re-engineered for a sustainable future. *Nucleic Acids Research*, page
1141 gkaf1271, 2025.
- 1142 [69] James Malone, Ele Holloway, Tomasz Adamusiak, Misha Kapushesky, Jie Zheng, Nikolay
1143 Kolesnikov, Anna Zhukova, Alvis Brazma, and Helen Parkinson. Modeling sample variables
1144 with an experimental factor ontology. *Bioinformatics*, 26(8):1112–1118, 2010.
- 1145 [70] Aarne Mämmelä, Jukka Riekk, and Markku Kiviranta. Loose coupling: An invisible thread
1146 in the history of technology. *IEEE Access*, 11:59456–59482, 2023.
- 1147 [71] Robert C Martin and Micah Martin. *Agile principles, patterns, and practices in C*. Pearson
1148 Education, 2006.
- 1149 [72] Tomas Mikolov, Kai Chen, Greg Corrado, and Jeffrey Dean. Efficient estimation of word
1150 representations in vector space. *arXiv preprint arXiv:1301.3781*, 2013.
- 1151 [73] Niklas Muennighoff, Nouamane Tazi, Loïc Magne, and Nils Reimers. Mteb: Massive text
1152 embedding benchmark. *arXiv preprint arXiv:2210.07316*, 2022.
- 1153 [74] Roberto Navigli and Paola Velardi. Learning domain ontologies from document warehouses
1154 and dedicated web sites. *Computational Linguistics*, 30(2):151–179, 2004.
- 1155 [75] Jianmo Ni, Chen Qu, Jing Lu, Zhuyun Dai, Gustavo Hernandez Abrego, Ji Ma, Vincent
1156 Zhao, Yi Luan, Keith Hall, Ming-Wei Chang, et al. Large dual encoders are generalizable
1157 retrievers. In *Proceedings of the 2022 Conference on Empirical Methods in Natural Language*
1158 *Processing*, pages 9844–9855, 2022.
- 1159 [76] Sebastian Mateos Nicolajsen. Extensibility in programming languages: An overview. *CoRR*,
1160 pages 1–12, 2025.
- 1161 [77] Beatrix Miranda Ginn Nielsen and Lars Kai Hansen. Hubness reduction improves sentence-
1162 bert semantic spaces. In *Northern Lights Deep Learning Conference*, pages 181–204. PMLR,
1163 2024.
- 1164 [78] Ian Niles and Adam Pease. Towards a standard upper ontology. In *Proceedings of the in-*
1165 *ternational conference on Formal Ontology in Information Systems-Volume 2001*, pages 2–9,
1166 2001.
- 1167 [79] Matthias Noback. The interface segregation principle. In *Principles of Package Design:*
1168 *Creating Reusable Software Components*, pages 55–66. Springer, 2018.
- 1169 [80] Natalya F Noy, Nigam H Shah, Patricia L Whetzel, Benjamin Dai, Michael Dorf, Nicholas
1170 Griffith, Clement Jonquet, Daniel L Rubin, Margaret-Anne Storey, Christopher G Chute,
1171 et al. Bioportal: ontologies and integrated data resources at the click of a mouse. *Nucleic*
1172 *acids research*, 37(suppl_2):W170–W173, 2009.
- 1173 [81] Zach Nussbaum and Brandon Duderstadt. Training sparse mixture of experts text embedding
1174 models. *arXiv preprint arXiv:2502.07972*, 2025.

- 1175 [82] Zach Nussbaum, John X Morris, Brandon Duderstadt, and Andriy Mulyar. Nomic embed:
1176 Training a reproducible long context text embedder. *arXiv preprint arXiv:2402.01613*, 2024.
- 1177 [83] Zach Nussbaum, John X Morris, Brandon Duderstadt, and Andriy Mulyar. Nomic embed:
1178 Training a reproducible long context text embedder. *arXiv preprint arXiv:2402.01613*, 2024.
- 1179 [84] Andrea Giovanni Nuzzolese, Anna Lisa Gentile, Valentina Presutti, and Aldo Gangemi. Se-
1180 mantic web conference ontology - a refactoring solution. In Harald Sack, Giuseppe Rizzo,
1181 Nadine Steinmetz, Dunja Mladenić, Sören Auer, and Christoph Lange, editors, *The Semantic*
1182 *Web*, pages 84–87, Cham, 2016. Springer International Publishing.
- 1183 [85] J Douglas Orton and Karl E Weick. Loosely coupled systems: A reconceptualization.
1184 *Academy of management review*, 15(2):203–223, 1990.
- 1185 [86] Francesco Osborne and Enrico Motta. Klink-2: integrating multiple web sources to gener-
1186 ate semantic topic networks. In *International Semantic Web Conference*, pages 408–424.
1187 Springer, 2015.
- 1188 [87] F. Pedregosa, G. Varoquaux, A. Gramfort, V. Michel, B. Thirion, O. Grisel, M. Blondel,
1189 P. Prettenhofer, R. Weiss, V. Dubourg, J. Vanderplas, A. Passos, D. Cournapeau, M. Brucher,
1190 M. Perrot, and E. Duchesnay. Scikit-learn: Machine learning in Python. *Journal of Machine*
1191 *Learning Research*, 12:2825–2830, 2011.
- 1192 [88] Yixin Peng, Yongli Mou, Bozhen Zhu, Sulayman Sowe, and Stefan Decker. Rwth-dbis at
1193 llms4ol 2024 tasks a and b: Knowledge-enhanced domain-specific continual learning and
1194 prompt-tuning of large language models for ontology learning. In *Open Conference Proceed-*
1195 *ings*, volume 4, pages 49–63, 2024.
- 1196 [89] Jeffrey Pennington, Richard Socher, and Christopher D Manning. Glove: Global vectors for
1197 word representation. In *Proceedings of the 2014 conference on empirical methods in natural*
1198 *language processing (EMNLP)*, pages 1532–1543, 2014.
- 1199 [90] Rashin Rahnamoun and Mehrnoush Shamsfard. Sbu-nlp at llms4ol 2025 tasks a, b, and
1200 c: Stage-wise ontology construction through llms without any training procedure. In *Open*
1201 *Conference Proceedings*, volume 6, 2025.
- 1202 [91] Balashanmuga Priyan Rajamohan, Alexander C Harding Bradley, Van D Tran, Jonathan E
1203 Gordon, Hayden W Caldwell, Redad Mehdi, Gabriel Ponon, Quynh D Tran, Ozan Dernek,
1204 Jarod Kaltenbaugh, et al. Materials data science ontology (mds-onto): Unifying domain
1205 knowledge in materials and applied data science. *Scientific Data*, 12(1):628, 2025.
- 1206 [92] Rob Raskin and Michael Pan. Semantic web for earth and environmental terminology (sweet).
1207 In *Proc. of the Workshop on Semantic Web Technologies for Searching and Retrieving Scien-*
1208 *tific Data*, volume 25, 2003.
- 1209 [93] Anton Razzhigaev, Matvey Mikhalechuk, Elizaveta Goncharova, Ivan Oseledets, Denis Dim-
1210 itrov, and Andrey Kuznetsov. The shape of learning: Anisotropy and intrinsic dimensions
1211 in transformer-based models. In *Findings of the Association for Computational Linguistics:*
1212 *EACL 2024*, pages 868–874, 2024.
- 1213 [94] Nils Reimers and Iryna Gurevych. Sentence-BERT: Sentence embeddings using Siamese
1214 BERT-networks. In Kentaro Inui, Jing Jiang, Vincent Ng, and Xiaojun Wan, editors, *Pro-*
1215 *ceedings of the 2019 Conference on Empirical Methods in Natural Language Processing and*
1216 *the 9th International Joint Conference on Natural Language Processing (EMNLP-IJCNLP)*,
1217 pages 3982–3992, Hong Kong, China, November 2019. Association for Computational Lin-
1218 guistics.
- 1219 [95] Nils Reimers and Iryna Gurevych. Sentence-bert: Sentence embeddings using siamese bert-
1220 networks. In *Proceedings of the 2019 Conference on Empirical Methods in Natural Language*
1221 *Processing*. Association for Computational Linguistics, 11 2019.
- 1222 [96] Chris Rupp and The SOPHISTS. *Requirements-Engineering UND -Management: Das Hand-*
1223 *buch für Anforderungen in jeder Situation*. Hanser, 2021.

- [97] Susanna-Assunta Sansone, Peter McQuilton, Philippe Rocca-Serra, Alejandra Gonzalez-Beltran, Massimiliano Izzo, Allyson L Lister, Milo Thurston, and FAIRsharing Community. Fairsharing as a community approach to standards, repositories and policies. *Nature biotechnology*, 37(4):358–367, 2019.
- [98] Marion Schaeffer, Matthias Sesboué, Jean-Philippe Kotowicz, Nicolas Delestre, and Cecilia Zanni-Merk. Olaf: An ontology learning applied framework. *Procedia Computer Science*, 225:2106–2115, 2023.
- [99] Wes Schafer, Oliver He, Anna L. Dunn, and Zachary E. X. Dance. Ontology for process chemistry – giving context to instrument data structured by the allotrope data model. Allotrope Connect Virtual Conference, April 2021. Virtual conference, April 19–26, 2021.
- [100] Schema.org Community Group. Schema.org. <https://schema.org/>, 2011.
- [101] Lukas Schmelzeisen and Steffen Staab. Learning taxonomies of concepts and not words using contextualized word representations: A position paper. *arXiv preprint arXiv:1902.02169*, 2019.
- [102] Lynn M Schriml, Elvira Mittra, James Munro, Becky Tauber, Mike Schor, Lance Nickle, Victor Felix, Linda Jeng, Cynthia Bearer, Richard Lichenstein, Katharine Bisordi, Nicole Campion, Brooke Hyman, David Kurland, Connor Patrick Oates, Siobhan Kibbey, Poorna Sreekumar, Chris Le, Michelle Giglio, and Carol Greene. Human disease ontology 2018 update: classification, content and workflow expansion. *Nucleic Acids Research*, 47(D1):D955–D962, 11 2018.
- [103] Semantic Web Company. Poolparty extractor – graph-based text mining at the highest level. <https://www.poolparty.biz/poolparty-extractor>. Accessed: 2026-01-27. PoolParty Extractor is an intelligent semantic text mining tool that combines natural language processing and machine learning with knowledge graph-based concept extraction to analyze and enrich unstructured text.
- [104] Aditya Sinha, Siqi Zeng, Makoto Yamada, and Han Zhao. Learning structured representations with hyperbolic embeddings. *Advances in Neural Information Processing Systems*, 37:91220–91259, 2024.
- [105] Barry Smith, Michael Ashburner, Cornelius Rosse, Jonathan Bard, William Bug, Werner Ceusters, Louis J Goldberg, Karen Eilbeck, Amelia Ireland, Christopher J Mungall, et al. The obo foundry: coordinated evolution of ontologies to support biomedical data integration. *Nature biotechnology*, 25(11):1251–1255, 2007.
- [106] Philip Strömert, Giacomo Lanza, Johannes Hunold, Steffen Neumann, et al. The Vibrational Spectroscopy Ontology (VIBSO), 2023.
- [107] Rudi Studer, V.Richard Benjamins, and Dieter Fensel. Knowledge engineering: Principles and methods. *Data Knowledge Engineering*, 25(1):161–197, 1998.
- [108] Imma Subirats-Coll, Kristin Kolshus, Andrea Turbati, Armando Stellato, Esther Mietzsch, Daniel Martini, and Marcia Zeng. Agrovoc: The linked data concept hub for food and agriculture. *Computers and Electronics in Agriculture*, 196:105965, 2022.
- [109] Qwen Team. Qwen2.5: A party of foundation models, September 2024.
- [110] The SOPHISTS. *Requirements Engineering - A Short RE Primer*. 2016.
- [111] TIB – Leibniz Information Centre for Science and Technology and University Library. Tib terminology service. <https://terminology.tib.eu/ts>. Accessed: 2025-05-02.
- [112] Hugo Touvron, Thibaut Lavril, Gautier Izacard, Xavier Martinet, Marie-Anne Lachaux, Timothée Lacroix, Baptiste Rozière, Naman Goyal, Eric Hambro, Faisal Azhar, et al. Llama: Open and efficient foundation language models. *arXiv preprint arXiv:2302.13971*, 2023.
- [113] Pierre-Yves Vandenbussche, Ghislain A Atezing, María Poveda-Villalón, and Bernard Vatan. Linked open vocabularies (lov): a gateway to reusable semantic vocabularies on the web. *Semantic Web*, 8(3):437–452, 2016.

- [114] Paola Velardi, Stefano Faralli, and Roberto Navigli. Ontolearn reloaded: A graph-based algorithm for taxonomy induction. *Computational Linguistics*, 39(3):665–707, 09 2013.
- [115] Henrique Schechter Vera, Sahil Dua, Biao Zhang, Daniel Salz, Ryan Mullins, Sindhu Raghuram Panyam, Sara Smoot, Iftekhar Naim, Joe Zou, Feiyang Chen, et al. Embeddinggemma: Powerful and lightweight text representations. *arXiv preprint arXiv:2509.20354*, 2025.
- [116] Stuart L Weibel and Traugott Koch. The dublin core metadata initiative. *D-lib magazine*, 6(12):1082–9873, 2000.
- [117] Patricia L Whetzel, Natalya F Noy, Nigam H Shah, Paul R Alexander, Csongor Nyulas, Tania Tudorache, and Mark A Musen. Biportal: enhanced functionality via new web services from the national center for biomedical ontology to access and use ontologies in software applications. *Nucleic acids research*, 39(suppl_2):W541–W545, 2011.
- [118] Thomas Wolf, Lysandre Debut, Victor Sanh, Julien Chaumond, Clement Delangue, Anthony Moi, Pierric Cistac, Tim Rault, Remi Louf, Morgan Funtowicz, Joe Davison, Sam Shleifer, Patrick von Platen, Clara Ma, Yacine Jernite, Julien Plu, Canwen Xu, Teven Le Scao, Sylvain Gugger, Mariama Drame, Quentin Lhoest, and Alexander Rush. Transformers: State-of-the-art natural language processing. In Qun Liu and David Schlangen, editors, *Proceedings of the 2020 Conference on Empirical Methods in Natural Language Processing: System Demonstrations*, pages 38–45, Online, October 2020. Association for Computational Linguistics.
- [119] Wentao Wu, Hongsong Li, Haixun Wang, and Kenny Q Zhu. Probbase: A probabilistic taxonomy for text understanding. In *Proceedings of the 2012 ACM SIGMOD international conference on management of data*, pages 481–492, 2012.
- [120] An Yang, Anfeng Li, Baosong Yang, Beichen Zhang, Binyuan Hui, Bo Zheng, Bowen Yu, Chang Gao, Chengen Huang, Chenxu Lv, et al. Qwen3 technical report. *arXiv preprint arXiv:2505.09388*, 2025.
- [121] Hui Yang and Jiaoyan Chen. Achieving hyperbolic-like expressiveness with arbitrary euclidean regions: A new approach to hierarchical embeddings, 2025.
- [122] Antonio Zaitoun, Tomer Sagi, and Mor Peleg. Generating ontology-learning training-data through verbalization. *Proceedings of the AAAI Symposium Series*, 4(1):233–241, Nov. 2024.
- [123] Bohui Zhang, Valentina Anita Carriero, Katrin Schreiberhuber, Stefani Tsaneva, Lucía Sánchez González, Jongmo Kim, and Jacopo de Berardinis. Ontochat: a framework for conversational ontology engineering using language models. In *European Semantic Web Conference*, pages 102–121. Springer, 2024.
- [124] Yanzhao Zhang, Mingxin Li, Dingkun Long, Xin Zhang, Huan Lin, Baosong Yang, Pengjun Xie, An Yang, Dayiheng Liu, Junyang Lin, et al. Qwen3 embedding: Advancing text embedding and reranking through foundation models. *arXiv preprint arXiv:2506.05176*, 2025.
- [125] Jingwei Zuo, Maksim Velikanov, Ilyas Chahed, Younes Belkada, Dhia Eddine Rhayem, Guillaume Kunsch, Hakim Hacid, Hamza Yous, Brahim Farhat, Ibrahim Khadraoui, et al. Falcon-h1: A family of hybrid-head language models redefining efficiency and performance. *arXiv preprint arXiv:2507.22448*, 2025.

1312 A Related Works

1313 Ontology learning (OL) has progressed from manual efforts to automated pipelines supported by
 1314 specialized infrastructure, classical systems, and, more recently, LLMs. We organize related work
 1315 into three themes: (1) ontology infrastructure and repositories for reuse and interoperability, (2)
 1316 automated systems for extracting ontologies from text, and (3) the emerging role of LLMs. This
 1317 framing highlights OntoLearner’s distinct role as a benchmarking framework that unites the three
 1318 areas by enabling systematic evaluation and comparison across infrastructure, classical systems, and
 1319 LLM-based methods.

1320 A.1 Ontology Infrastructure and Repositories

1321 A mature ecosystem of platforms supports ontology creation, discovery, and reuse across scientific
1322 domains. BioPortal, developed by the National Center for Biomedical Ontology (NCBO), exempli-
1323 fies the ontology development and repository platform (OD category), hosting over 1,000 biomedical
1324 ontologies with rich search, visualization, and mapping tools [80, 117, 12]. The Open Biomedical
1325 Ontologies (OBO) Foundry takes a community-driven approach, coordinating over 200 interoper-
1326 able ontologies built on shared principles of openness and orthogonality to minimize redundancy
1327 [105]. Domain-specific portals like AgroPortal extend this model to agriculture and plant sciences
1328 by reusing the BioPortal infrastructure [54].

1329 Complementing development platforms are access and lookup services (OL category) that provide
1330 centralized, API-driven discovery. The EMBL-EBI Ontology Lookup Service (OLS) offers a unified
1331 REST API for querying around 270 OWL and OBO ontologies [56, 37], while the TIB Terminol-
1332 ogy Service extends this model beyond life sciences to engineering, chemistry, and physics domains
1333 [61, 111, 25]. Vocabulary integration services (VI category) focus on term reuse and alignment:
1334 the Linked Open Vocabularies (LOV) project curates metadata for commonly used RDF vocabular-
1335 ies [113], while the Unified Medical Language System (UMLS) integrates over 60 biomedical
1336 terminologies into a single Metathesaurus with more than 2 million terms [13]. Standards registries
1337 like FAIRsharing provide interlinked catalogs of community standards, databases, and ontologies to
1338 enhance visibility and FAIR compliance [97].

1339 While these platforms excel at hosting, discovery, or integration, the landscape remains fragmented
1340 with overlapping content and inconsistent modeling across ontologies [58]. More critically, none
1341 provides systematic support for *evaluating* how well AI models can learn, reconstruct, or enrich
1342 ontologies. Importantly, none provides end-to-end engineering support or systematic benchmarking
1343 of AI-driven ontology enrichment. Table 1a summarizes the strengths of each infrastructure category
1344 and highlights OntoLearner’s novel contribution as an OL support system for benchmarking and
1345 evaluation.

1346 A.2 Classical Ontology Learning Frameworks

1347 Classical OL systems established foundational approaches by combining natural language process-
1348 ing (NLP) with machine learning (ML) in modular pipelines, prior to the advent of large language
1349 models. Text2Onto [19] pioneered probabilistic ontology modeling by integrating linguistic pattern
1350 matching (e.g., Hearst patterns), statistical association measures (PMI, TF-IDF), and hierarchical
1351 clustering to extract terms and taxonomic relations. It introduced confidence scoring, supported
1352 incremental learning from evolving corpora, and featured a graphical user interface for interactive
1353 refinement. Velardi et al.’s OntoLearn [74] employed lexical patterns and syntactic parsing to induce
1354 taxonomies, with the Reloaded version [114] using graph-based pruning algorithms to construct co-
1355 herent tree hierarchies, successfully reconstructing parts of WordNet. OntoGen [34] demonstrated
1356 early human-in-the-loop design by combining unsupervised k-means clustering and TF-IDF weight-
1357 ing to suggest candidate concepts through a user-friendly editor. Its plug-in architecture and inter-
1358 active visualizations foreshadow many design choices in modern LLM-based ontology assistants.

1359 Larger-scale systems pushed the boundaries of automation and coverage. The Never-Ending Lan-
1360 guage Learner (NELL) [16] employed coupled semi-supervised learning using Naive Bayes classi-
1361 fiers, logistic regression, and bootstrapped extraction rules to continuously populate a seed ontology
1362 from web text. Over years, NELL accumulated millions of beliefs under knowledge coupling con-
1363 straints to reduce semantic drift. However, it did not support the invention of new classes or complex
1364 axioms. Klink-2 constructed the 14,000-topic Computer Science Ontology by analyzing keyword
1365 co-occurrence and citation networks in scholarly publications [86]. OLAF proposed a fully auto-
1366 mated pipeline using dependency parsing, frequency heuristics, and rule-based relation extraction to
1367 produce “minimum viable ontologies” with minimal human intervention [98].

1368 In industry, OL has been deployed at massive scale. Microsoft’s Probase mined 1.68 billion web
1369 pages using Hearst pattern extraction and statistical co-occurrence analysis to construct a 2.7-
1370 million-concept probabilistic taxonomy capturing relationship uncertainty [119]. Diffbot’s commer-
1371 cial Knowledge Graph integrates computer vision, NLP, and entity resolution to extract structured
1372 data from over 60 billion web pages into a proprietary ontology with 10+ billion entities [27]. Com-

1373 mercial systems like PoolParty combine NLP-based term extraction, named entity recognition, and
1374 graph-based disambiguation in enterprise-grade taxonomy engineering platforms [103].

1375 These classical systems demonstrate the viability of automated ontology construction, but most re-
1376 lied on deterministic pattern matching, shallow statistical features, or domain-specific heuristics.
1377 Their modular architectures and algorithmic innovations remain relevant to OntoLearner in three
1378 ways: as *baselines* for comparing LLM performance, as *hybrid components* that can augment LLM
1379 outputs (e.g., graph pruning to improve structural coherence), and as sources of *validation met-*
1380 *rics* (e.g., graph-based quality measures). OntoLearner’s planned Text2Onto module¹⁶ extends this
1381 lineage by generating synthetic training corpora from existing ontologies, enabling systematic eval-
1382 uation of how well LLMs can replicate classical pattern-based extraction pipelines.

1383 A.3 LLMs for Ontology Learning

1384 The emergence of LLMs has introduced new paradigms for OL, offering flexible pipelines for con-
1385 cept extraction, taxonomy induction, and interactive modeling. However, recent evaluations reveal
1386 both promise and limitations. NeOn-GPT integrates structured prompt engineering sequences with
1387 the NeOn Methodology to elicit concepts, hierarchies, and OWL axioms. Evaluations on a gold-
1388 standard Wine ontology found that while the model could produce plausible structures, it struggled
1389 with logical consistency and reasoning, necessitating iterative human-in-the-loop refinement [32].
1390 OntoGPT applies schema-based extraction via the SPIRES method in biomedical domains, using
1391 recursive zero-shot prompting grounded in existing ontology identifiers [17]. OntoChat comple-
1392 ments these by introducing conversational interfaces that parse intent and dialogue states to guide
1393 early-stage modeling tasks such as competency question generation [123].

1394 A more radical approach is OLLM, which forgoes modular design in favor of end-to-end fine-tuning:
1395 a language model is trained to generate entire taxonomies from domain corpora using custom reg-
1396 ularizers and embedding-based graph metrics. Compared to pipelined methods, OLLM achieves
1397 higher semantic coherence on Wikipedia and arXiv-derived ontologies [65]. Complementary to
1398 these generative systems, Ontolearn [23] focuses on learning OWL class expressions over large-
1399 scale knowledge graphs. It implements symbolic and neuro-symbolic algorithms (e.g., EvoLearner
1400 [47], DRILL [24]) to induce class expressions, and adds an LLM-powered verbalization module
1401 to translate those expressions into natural language. While not an end-to-end OL system, On-
1402 tolearn contributes important tooling for downstream reasoning, explanation, and integration with
1403 SPARQL-based infrastructure.

1404 To address the lack of standardized evaluation settings across these systems, the LLMs4OL initia-
1405 tive we introduced formalizes the OL task space by benchmarking core subproblems—term typing,
1406 taxonomy discovery, and non-taxonomic relation extraction—across domains including medicine,
1407 geography, and lexico semantics [8]. Challenge submissions showed that hybrid pipelines combining
1408 semantic embeddings (e.g., BERT, mpnet) with few-shot LLM prompting consistently outperform
1409 single-method baselines, though performance varies by domain and task [38, 39]. Studies of hu-
1410 man-LLM collaboration report that LLMs can automate repetitive extraction steps and reduce effort
1411 by over 50%, but semantic drift and inconsistencies persist without human oversight [60].

1412 Table 1b contrasts these LLM-based systems and highlights a key distinction: most are *construction*
1413 *tools* focused on building ontologies, whereas OntoLearner is a *benchmarking framework* designed
1414 to evaluate and compare OL methods. It directly implements and extends the LLMs4OL paradigm
1415 by providing standardized datasets, evaluation metrics, and reproducible pipelines across 177 on-
1416 tologies in 22 domains—far exceeding prior efforts. This enables researchers to systematically
1417 assess methods such as NeOn-GPT’s structured prompting, OntoGPT’s schema-based extraction,
1418 retrieval-augmented generation, or OLLM’s fine-tuning.

1419 OntoLearner distinguishes itself through three key contributions. First, its benchmarking-first design
1420 enables curated ontologies to be transformed into train/dev/test splits with automatic evaluation
1421 metrics, supporting rigorous model comparisons over ad hoc experimentation. Second, it bridges
1422 symbolic and neural approaches: classical algorithms serve as baselines; LLMs augment or replace
1423 modules; and hybrid workflows combine structured retrieval with generative modeling. Third, its
1424 modular, open-source architecture supports community-driven development under FAIR principles,
1425 with seamless integration of new ontologies via GitHub or HuggingFace. By advancing OL from

¹⁶https://ontolearner.readthedocs.io/package_reference/text2onto.html

¹⁴²⁶ exploratory prototyping to reproducible evaluation, OntoLearner provides essential infrastructure
¹⁴²⁷ for trustworthy, AI-assisted ontology engineering.

Supplementary Files

This is a list of supplementary files associated with this preprint. Click to download.

- [supplementarymaterials.pdf](#)