LLMs4OL 2025: The 2nd Large Language Models for Ontology Learning Challenge at the 24th ISWC
LLMs4OL 2025 Task Participant Long Papers
https://doi.org/10.52825/ocp.v6i.2895

Published: 01 Oct. 2025

IRIS at LLMs4OL 2025 Tasks B, C and D: Enhancing Ontology Learning Through Data Enrichment and Type Filtering

Insan-Aleksandr Latipov[1,*], Mike Holenderski[1], and Nirvana Meratnia[1]

[1]Department of Mathematics and Computer Science, Eindhoven University of Technology, The Netherlands

*Correspondence: Insan-Aleksandr Latipov, i.latipov@tue.nl;

Abstract. Ontology Learning (OL) automates extracting structured knowledge from unstructured data. We study how model-agnostic data manipulations can boost performance of Large Language Models (LLM) on three OL tasks, i.e., term typing, taxonomy discovery and non-taxonomic relation extraction, from the LLMs4OL 2025 Challenge. We investigate two *input-enrichment* techniques, i.e., (i) data augmentation, and (ii) addition of term and type definitions that expand the information supplied to an LLM. Complementing the enrichment techniques, we also study a pruning technique, i.e., a similarity-based candidate filtering technique that narrows the candidate space in taxonomy discovery and non-taxonomic relation extraction to the most semantically relevant types. When applied individually, each technique boosts precision–recall metrics over the vanilla setting where an LLM is trained on the original data. However, applied together they yield the best scores in five out of the seven ontology–task combinations, showing synergetic benefits. Our findings show that careful curation of inputs can itself yield substantial performance improvements. Codebase and all training artifacts are available at our GitHub repository[1].

Keywords: Ontology Learning, Large Language Model, Data Augmentation, Definition Mining, Similarity Filtering

1. Introduction

Ontology Learning (OL) aims to extract information from unstructured or semi-structured text into a formal ontology that machines can reason about [1]. Despite recent advances in information extraction models, for example described in Deng et al. [2], the OL problem still remains challenging. Accurately capturing and interpreting domain-specific language, often dense with specialized terminology, is central to ontology learning. When combined with data sparsity and extreme class imbalance, it requires substantial manual curation and deep domain expertise [3]. Recent advances in Large Language Models (LLMs) offer a concrete way to mitigate existing problems in OL. First, extensive pre-training equips LLMs with wide cross-domain lexical coverage, providing them

[1]`https://github.com/AFigaro/LLMs4OL_2025/tree/main`

with a comprehensive overview across a wide range of domains. Second, fine-tuning can specialize these models to a new domain, easing the bottleneck posed by highly technical terminology. Third, proper tuning of OL tasks allows an LLM to acquire new ontology concepts from only a handful of labeled examples, mitigating data sparsity and extreme class imbalance [4].

Ontology learning consists of four steps: corpus preparation, terminology extraction, conceptualisation (which is split into term typing, taxonomy discovery, non-taxonomic relation extraction), and axiom discovery [1]. The *Large Language Models for Ontology Learning* (LLMs4OL) challenge [1] benchmarks how well LLMs can perform on different OL tasks in few- and zero-shot settings. The second iteration of LLMs4OL 2025 examines the capabilities of LLMs in 4 OL subtasks [5]:

- **Subtask A**—*Text2Onto*: extract domain terms and types from raw texts;
- **Subtask B**—*Term Typing*: assign the most specific type to a given term;
- **Subtask C**—*Taxonomy Discovery*: predict taxonomic *is-a* links between types;
- **Subtask D**—*Non-Taxonomic Relation Extraction*: identify other semantic relations between type pairs.

All subtasks are officially scored with *Precision*, *Recall*, and *F1-score* using predefined training and test splits.

However, the advantages mentioned above of LLMs materialise only when the input provides sufficient context - exactly what the LLMs4OL Challenge deliberately withholds to stress-test models. Our own inspection of the LLMs4OL 2025 data revealed the three bottlenecks listed below:

- (i) **rare-class sparsity** — many types appear only once or twice in the training data;
- (ii) **context deficit** — terms and types are given without explanatory definitions, leaving the model to infer semantics from lexical form alone;
- (iii) **quadratic explosion** — a naive solution to subtasks C and D scores all type pairs to decide whether any relation holds has complexity $O(|T|^2)$, where T is a set of all types, even though almost all pairs do not have any relations.

The three outlined limitations suggest that an effective pipeline for OL must (i) augment training data, (ii) enrich the context-poor input with term and type definitions, and (iii) prune implausible type pairs for taxonomy discovery and non-taxonomic relation extraction. To address these limitations, we propose the following research question: To what extent do (i) data augmentation techniques, (ii) term and type definition mining, and (iii) candidate type filtering improve LLMs' capabilities for term typing, taxonomy discovery, and non-taxonomic relation extraction? To address this question, we decompose it into two task-specific research questions, since subtask B is dominated by a different set of bottlenecks than subtasks C and D:

- **RQ1** ***(Term Typing).*** To what extent do data augmentation and term definition mining improve precision, recall, and F1-score of an LLM-based classifier for term typing?
- **RQ2** ***(Taxonomy Discovery & Non-Taxonomic RE).*** How do similarity-based candidate filtering and type definition mining improve precision, recall, and F1-score of an LLM-based classifier for taxonomy discovery and non-taxonomic relation extraction?

To address our research questions, we concentrated on subtasks B, C, and D, as they collectively represent the conceptualisation phase of OL. Regarding ontologies, we decided to focus on biomedical investigations OBI [6], and Earth and material science ontologies presented by MatOnto [7] and SWEET [8]. We concentrate on OBI, MatOnto, and SWEET because the biomedicine, materials, and Earth science domains share deep hierarchies and highly specialised vocabularies, providing a coherent test case for our data enrichment study. Each subtask is evaluated independently of each other on different ontologies.

The rest of the paper is organised as follows. Section 2 presents an overview of the existing ontology learning approaches and positions our study within the recent LLM-based information extraction frameworks. Section 3 presents details of our methodology. Section 4 reports on exact data sources, licensing conditions, and preprocessing scripts to ensure full reproducibility. Section 5 reports precision, recall, and F1-score in every ablation required to answer RQ1 and RQ2 and analyses the results, highlighting the impact of each technique on data-scarce and imbalanced scenarios. Finally, Section 6 summarises the main findings, outlines limitations, and sketches avenues for extending LLM-assisted ontology learning in future work.

2. Related Work

Early ontology learning systems exploited rule-based or shallow statistical cues mined from corpora and thesaurus. Classic pipelines such as Text2Onto [9] combined lexico-syntactic patterns with TF–IDF heuristics to induce concepts and taxonomic links from domain text. Prior to the rise of LLM-based methods, automatic and semi-automatic OL pipelines relied on three main paradigms according to Asim et al. [10]: (i) **linguistic** (rule- or pattern-based) techniques that harness handcrafted lexicons, part-of-speech tags, and parsing; (ii) **statistical** techniques that mine corpus-level co-occurrence and frequency statistics of ontology terms and types; and (iii) **logical** techniques—most notably inductive logic programming—that induce formal axioms from previously extracted artifacts. All three paradigms are highly dependent on manually built resources and meticulous task-specific feature engineering, keeping domain experts in the loop.

LLMs inherit broad factual and common-sense knowledge from the data they are pre-trained on, can be guided with a handful of in-context examples, and even act as agents that can query, retrieve, or even self-generate supplementary evidence. All these traits can potentially address the data scarcity and context-deficit problems of OL stated above, as shown by Du et al. [11]. Recent case studies, for example, Lippolis et al.[12], showed that in some domains proprietary models are capable of producing ontologies of sufficient quality to meet the requirements of ontology engineers. However, relying on such proprietary systems is often impractical, given data-security concerns and the need to tailor models to specialised domains and project-specific requirements.

Unified information extraction (UIE) introduced by Lu et al. [13] demonstrated capabilities in various OL-related tasks such as the extraction of entities and relations by using a single structured extraction language and training a sequence-to-sequence model to generate the corresponding structured outputs based on that schema. This approach established a state-of-the-art baseline in 13 datasets but still relied on supervised fine-tuning, a process that demands heavy manual curation. Wang et al. [14] introduced a modification of UIE - called InstructUIE - to overcome the need for a fixed schema, using natural language instructions, pairing each prompt (task description + answer options list + input text) with the desired output. ChatUIE, introduced by Xu et al. [15], pushed the idea further. A ChatGLM backbone had three-stage training:

supervised fine-tuning, reward-model learning, and PPO reinforcement learning, plus a generation-constraint matrix that blocks tokens not present in the source, mitigating type confusion and sample imbalance across Named Entity Recognition (NER), Relation Extraction (RE), and Event Extraction (EE) datasets.

The LLMs4OL 2024 challenge explored a wide spectrum of LLMs - from compact BERT to proprietary GPT-3.5 and open source LLaMA-3-70B - combined with strategies such as lightweight fine-tuning, retrieval-augmented prompting, and prompt engineering [4]. These approaches achieved strong precision, recall, and macro-F1 on several ontologies, yet they left certain domains only partially resolved, underscoring the need for systematic, model-agnostic add-ons such as data augmentation, definition enrichment, and similarity-based filtering.

3. Methodology

This section explains the details of our solutions for subtasks B-D. We begin by formally defining the three challenge subtasks, i.e., term typing (B), taxonomy discovery (C), and non-taxonomic relation extraction (D), so that the notation is fixed for the remainder of the paper. We also describe the training and test data for these subtasks. This section is divided into two parts according to the research questions. Section 3.1 presents Subtask B, by first describing the baseline solution and then the two data enrichment techniques (i.e., data augmentation and term and type definition mining). Section 3.2 treats Subtasks C and D together, covering the baseline solution first and then describing similarity-based candidate filtering.

The term typing can be formally defined as the following. Let $\mathcal{T}$ be the set of ontology-specific types and let a natural language text x denote an input term string. Term typing asks a model to output a subset $Y \subseteq \mathcal{T}$ that contains all types applicable to x. The term typing training data provides only term-to-type mappings, with no accompanying definitions, and the test set consists solely of term strings.

The taxonomy discovery and non-taxonomic relation extraction ask whether a specified semantic relation holds for a given pair of ontology types. For taxonomy discovery, the relation is fixed: given an ordered pair of child and parent, denoted by $c, p \in \mathcal{T}$, decide whether the taxonomic link *isChildOf*(c, p) is true. For non-taxonomic relation extraction, the relation is part of the query: given a triple for head, relation, and tail denoted by (h, r, t) with $h, t \in \mathcal{T}$ and $r \in \mathcal{R}$, where $\mathcal{R}$ is a set of all possible relations, determine whether the relation $r(h, t)$ holds. The training data for these subtasks provides a list of all training types, without any definitions, a list of all type pairs/triples, and a list of all training relations for the non-taxonomic relation extraction. The test data consists of a list of all types and, for non-taxonomic relation extraction, a list of test relations.

3.1 Subtask B: Term Typing

We consider term typing as multi-label classification. For every type $t \in \mathcal{T}$, the model decides independently of the other types whether t applies to the input term. For a baseline solution, we only use the original challenge term–type pairs without any data augmentation or term definitions. Further, we add two data enrichment techniques to the baseline solution: data augmentation for additional term-to-type mappings and term definition mining for additional context. Figure 1 shows the overall workflow for term typing that combines these two enrichment techniques into a single pipeline. The pipeline augments the original training data with additional term-to-type mappings and enriches each term with automatically mined definitions before fine-tuning a DeBERTa-v3-large classifier introduced by He et al. [16]. A development set is used to calibrate the decision threshold, which determines the minimum prediction probability required for a type to be assigned to a term during inference. The following subsections describe each component in detail.

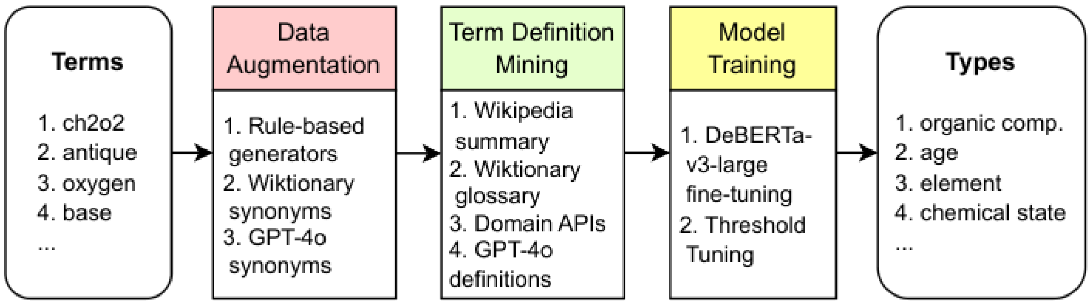

Figure 1. *Overview of the term typing pipeline. First, the* ***data augmentation*** *step adds additional term-to-type mappings. Then,* ***term definition mining*** *enriches each term with automatically mined definitions. Finally, the DeBERTa-v3-large classifier is fine-tuned on term-to-type mappings and threshold is tuned on held-out set.*

3.1.1 Baseline Solution

Input encoding. We use a DeBERTa-v3-large encoder and fine-tune its parameters rather than keeping the encoder frozen. We choose the DeBERTa model because it balances performance and computational cost and has strong benchmarks in language understanding tasks. Although this choice is purely plug-and-play and has proven effective in our setting, a comparison with domain-specific models (e.g., BioBERT or SciBERT) would be a valuable direction for future work to assess potential gains in specialised domains. Each input term string x is fed into the encoder, which outputs a hidden vector $h \in \mathbb{R}^d$. This vector is then passed through a task-specific linear layer that produces logits, one per ontology type:

$$L(x) = \big(L_1, \dots, L_{|\mathcal{T}|}\big) \in \mathbb{R}^{|\mathcal{T}|}, \tag{1}$$

Probability layer and decision rule. Sigmoid activations give posteriors per type $\sigma_i = \sigma\big(L_i\big)$. Collecting the posteriors of each type in a vector $\boldsymbol{\sigma}(x) = (\sigma_1, \dots, \sigma_{|\mathcal{T}|})$, we obtain a prediction vector for any threshold τ (by default $\tau = 0.5$) by using the element-wise rule

$$\hat{\mathbf{y}}(x) \;=\; \big[\,\mathbb{1}\{\sigma_i(x) \geq \tau\}\,\big]_{i=1}^{|\mathcal{T}|} \;\in\; \{0,1\}^{|\mathcal{T}|}. \tag{2}$$

Because the number of types that can apply to a term varies across ontologies—some assign several types, others only one or two, we calibrate the probability threshold on a held-out development set. To ensure that the calibration reflects the true label distribution, the development set is constructed to preserve the overall frequency of each type, so that both common and rare types are proportionally represented. In cases where some labels occurred too infrequently to allow stratified splitting, we used random splitting while ensuring representative coverage of the frequent types. Specifically, we hold out 10% of the training data for threshold calibration only and do not include it in any further training. This setup helps prevent information leakage and ensures that the selected threshold generalizes to unseen data:

$$\tau^* \;=\; \underset{\tau \in \{0.05, 0.10, \dots, 0.50\}}{\arg\max} \quad \text{F1-score}\big(\hat{\mathbf{y}}_{dev}, y_{\text{dev}}\big). \tag{3}$$

Here, $\hat{\mathbf{y}}_{dev} \in [0,1]^{m\times|\mathcal{T}|}$ is the matrix of predictions for the m terms from the development split, and $y_{\text{dev}} \in \{0,1\}^{m\times|\mathcal{T}|}$ is the corresponding ground-truth label matrix.

At test time, every type whose probability exceeds τ^* is assigned to the term; if none do, the type with the highest probability is returned so that each term receives at least one label.

Loss and optimisation. For a batch of B terms, we form the binary label matrix $Y \in \{0,1\}^{B\times|\mathcal{T}|}$, where $y_{bi} = 1$ if the b-th term is assigned the i-th type, and 0 otherwise. We then minimize a *class-weighted* binary cross-entropy loss:

$$\mathcal{L} = \frac{1}{B}\sum_{b=1}^{B}\sum_{i=1}^{|\mathcal{T}|} w_i[y_{bi}\log\sigma_{bi} + (1-y_{bi})\log(1-\sigma_{bi})], \qquad w_i = \frac{N-f_i}{f_i}, \tag{4}$$

where f_i is the number of positive occurrences of type i in the training data and N the total sample count, so that rare classes receive a larger gradient weight. Hyper-parameters for term typing were shared across all experiments. These parameters are AdamW optimiser, learning rate 2×10^{-5}, batch size 16, max length 512 if definitions enabled, else 128, and 10 epochs.

3.1.2 Data Augmentation

The original training splits are extremely small - many types have no more than two positive examples — so the baseline model suffers from both rare-class sparsity and limited lexical variety. We therefore enlarge the data by generating additional entries for every training term using the methods explained below. This increases the size of the dataset without any manual annotation and gives the encoder more lexical evidence from which to learn.

A manual inspection of the OBI, MatOnto, and SWEET ontologies revealed recurring lexical patterns specific to each domain (e.g. SI units in all ontologies, chemical formulae in SWEET, company suffixes, and biomedical acronyms in OBI). Based on these observations, we created a small catalogue of *rule-based generators* that rewrote a term while preserving its meaning. Only rules relevant to a given ontology were activated, so unnecessary noise is avoided. In addition to rule-based techniques, we use Wiktionary to collect synonyms for each term and the GPT-4o API [17] to generate additional lexical variants. For every term in the original training split, we apply these methods and add all newly obtained terms to the training data, assigning them the same type labels as the original term. The menu of augmentation techniques is listed below, and their ontology coverage is summarised in Table 1; global statistics before and after augmentation appear in Table 2.

- **Rule-based generators**
  - *Unit swaps* (MatOnto, SWEET, OBI) — swaps written-out units and SI symbols using manual written map: natural language name $\leftrightarrow$ SI unit; *metres per second* $\leftrightarrow$ `m/s`; *square metre* $\leftrightarrow$ $\mathtt{m}^2$.
  - *Chemical swaps* (SWEET) — replaces chemical names with formulas and vice-versa using molecular name $\leftrightarrow$ molecular formula map; `carbon dioxide` $\leftrightarrow$ $\mathtt{CO}_2$; `ozone` $\leftrightarrow$ $\mathtt{O}_3$.
  - *Simple tweaks* (MatOnto) — simple spelling changes: plural $\leftrightarrow$ singular (`crystals` $\leftrightarrow$ `crystal`); case variants (`Basalt` $\leftrightarrow$ `basalt`); hyphen $\leftrightarrow$ space (`iron-ore` $\leftrightarrow$ `iron ore`).

  - *Organisation & number rewrites* (OBI) — company-suffix removal and Roman $\leftrightarrow$ Arabic numbers swap; `Advanced Instruments, Inc.` $\leftrightarrow$ `Advanced Instruments`; `Phase II` $\leftrightarrow$ `Phase 2`.
  - *Acronym map* (OBI) — map biomedical acronyms and what they stand for; `OWL` $\leftrightarrow$ `Web Ontology Language`; `FCS` $\leftrightarrow$ `fetal calf serum`.
- **Wiktionary synonyms** (all) — up to ten short synonyms for a term from the Wiktionary, if available (`hurricane` $\rightarrow$ `cyclone`).
- **GPT-4o synonyms** (all) — prompted OpenAI GPT-4o model with prompt: "*List up to 3 short English synonyms ($\leq$ 3 words) for 'X'.*" e.g. `diesel particulate matter` $\rightarrow$ `diesel soot`.

Table 1. *Which augmentation modules are applied to each ontology.*

Ontology	Unit	Chem	Tweaks	Org./Numb./Acr.	Wikt.	GPT
MatOnto	✓	—	✓	—	✓	✓
OBI	✓	—	—	✓	✓	✓
SWEET	✓	✓	—	—	✓	✓

Table 2. *Training–set sizes before and after augmentation.*

Ontology	Original	Augmented
MatOnto	85	544
OBI	201	835
SWEET	1.558	9.158

To evaluate how the data augmentation impacts term typing, we simply replace the training file with the augmented one; the model, loss, hyperparameters, and threshold calibration described in Section 3.1.1 remain unchanged.

3.1.3 Automatic Definition Mining (Terms & Types)

All inputs (whether terms for subtask B or types for subtasks C and D) are context-free. This means that a bare term must be mapped to its correct ontology type(s), or two types must be linked to each other. By adding a short definition, we expose an LLM encoder to learn not only terms and types but also their meaning. During training, this extra context helps the model learn finer distinctions among similar strings; at test time, it lets the model compare new terms and types to previously seen ones through their textual descriptions.

In our study, we do not make a distinction between the definitions of mining for terms and for types. The same pipeline applies uniformly across all subtasks and ontologies, as both cases involve retrieving a short informative definition for a single string - regardless of whether that string denotes a term or a type — using the same sources, query logic, and integration method. The LLMs4OL challenge rules explicitly forbid using the reference ontologies themselves (or any artifacts derived from the ontology) for external knowledge. Our mining pipeline therefore queries only open, non-ontology resources - general encyclopedias, lexical dictionaries, and domain-specific public APIs — so that every definition is legally obtained and reproducible.

We use a simple, fixed-order list of sources for querying - given an input string, we consecutively query the list below until the definition is retrieved. Wikipedia comes first because its concise lead sentences cover most labels, are easy to find in bulk, and convey the core meaning without unnecessary details. If Wikipedia has no entry, we try Wiktionary as it has a wider coverage, and only then the more specialised domain APIs, whose answers can be longer or sometimes inconsistent. More specifically, our term and type definition mining pipeline looks as follows:

- **Wikipedia summary** The first sentence of a lead paragraph extracted using the REST endpoint `/page/summary/⟨term⟩`.
- **Wiktionary glossary** First English definition line returned by the Wiktionary API `/page/definition/`⟨term⟩.
- **Domain APIs** (ontology-specific)
  - OBI: *MeSH* scope notes[2], *PubChem* compound descriptions[3], *UniProt* protein-function lines[4].
  - SWEET & MatOnto: *USGS* mineral glossary[5], *NOAA* weather glossary[6], *ADS* "Universal glossary" abstracts[7].
- **GPT-4o fallback** If all web sources fail we use OpenAI API for GPT-4o model to generate a definition, e.g. *"Give a one-sentence definition of the earth-science term '⟨X⟩'."* This synthesises a concise description while avoiding ontology jargon.

When we study the impact of definitions in term typing, we add the mined definitions to the term, separated by `[DEF]` token, e.g.:

```
[TERM] diesel particulate matter [DEF] Soot and other fine particles produced by diesel
engines.
```

Definition mining is compatible with data augmentation. If we use both of them together, we attach the same mined definition to the original term and to every synonym, abbreviation, or reformatted spelling produced by the augmentation script. This yields two complementary signals: lexical variety from the new surfaces and semantic grounding from the shared glossary. We expect the combination to be especially helpful for rare terms, where the model now sees multiple spellings and an informative definition during training and at inference time. Other components of the training pipeline are not modified.

3.2 Subtasks C (Taxonomy Discovery) & D (Non-Taxonomic Relation Extraction)

We treat both taxonomy discovery and non-taxonomic relation extraction subtasks as binary classification. For taxonomy discovery, the label is $y = 1$ if the taxonomic link *isChildOf*(c, p) holds for a given pair (c, p) and $y = 0$ otherwise; for the non-taxonomic relation extraction, the label is $y = 1$ if the non-taxonomic relation $r(h, t)$ holds for a triple (h, r, t) and $y = 0$ otherwise.

For taxonomy discovery and non-taxonomic relation extraction, we cannot use the setting of multi-label classification, as we did in term typing. In term typing, we predict from a fixed label set $\mathcal{T}$, so a single multi-label head is natural. Here, the set of valid answers depends on the query. For illustration, consider taxonomy discovery. For a given

[2]clinicaltables.nlm.nih.gov
[3]pubchem.ncbi.nlm.nih.gov
[4]uniprot.org
[5]mrdata.usgs.gov
[6]forecast.weather.gov/glossary.php
[7]ui.adsabs.harvard.edu

child type, any of the remaining $|\mathcal{T}| - 1$ types can serve as a potential parent. When we examine a different child, its candidate-parent set likewise includes the first child we analysed. A global multi-label layer would therefore either need (i) a separate output vector for every query type, or (ii) an enormous, mostly zero vector that enumerates every possible pair or triple - both of which are impractical and heavily skewed toward negatives. By scoring each candidate pair or triple independently, we keep a single, compact output (two logits), while still allowing the model to handle unseen types at test time.

Figure 2 presents the complete workflow for these subtasks. Both of them share the same pipeline except for the output format (pairs vs. triples). The baseline approach trains a binary DeBERTa-v3-large classifier on positive and negative examples of linked types and applies threshold tuning on a development set. To further improve performance, we incorporate two enrichment strategies: (i) similarity-based candidate filtering, which restricts the search space to the K most semantically similar types using sentence embeddings and nearest-neighbour retrieval; and (ii) type definition mining, which injects short textual descriptions from external sources. The next subsections describe each component in detail.

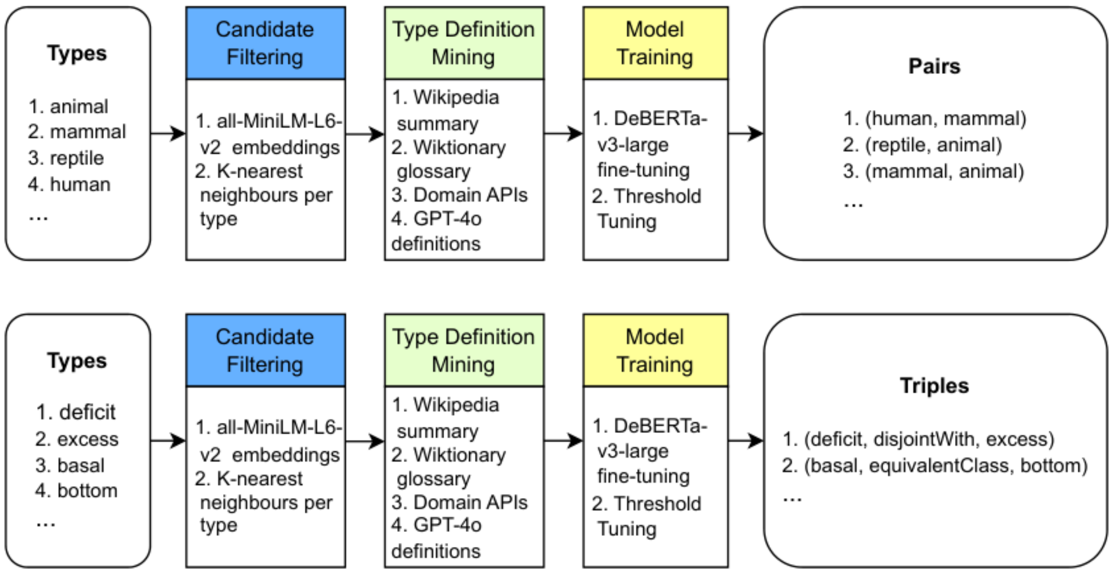

Figure 2. *Workflow for Taxonomy discovery and Non-taxonomic Relation Extraction. Unlike Term typing, both these subtasks are provided with raw types as input. Both pipelines use* ***similarity-based candidate filtering*** *to reduce the search space for potential pairs/triples and apply* ***type definition mining*** *technique to enrich type representations before fine-tuning a DeBERTa-v3-large classifier and calibrating thresholds. The only one difference between tasks is the output-pairs of isChildOf relation for taxonomy discovery and triples for non-taxonomic relation extraction.*

3.2.1 Baseline Solutions

Input encoding and decision rule. We again fine-tune the DeBERTa-v3-large model. However, this time, inputs are wrapped in a tagged string as follows:

$$\begin{aligned}\text{Task C:}\quad & \texttt{[CHILD]}\ c\ \big(\texttt{[DEF]}\ d_c\big)\ \texttt{[PARENT]}\ p\ \big(\texttt{[DEF]}\ d_p\big)\\ \text{Task D:}\quad & \texttt{[HEAD]}\ h\ \big(\texttt{[DEF]}\ d_h\big)\ \texttt{[RELATION]}\ r\\ & \texttt{[TAIL]}\ t\ \big(\texttt{[DEF]}\ d_t\big),\end{aligned}$$

where each [DEF] line is optional and provides the definition d of the corresponding type (see Section 3.1.3). The encoder output vector feeds a two-logit head $L = (L_0, L_1) \in \mathbb{R}^2$.

Soft-max gives class posteriors $\sigma_c = \exp(L_c)/\sum_j \exp(L_j)$. We obtain a prediction for a given input string by comparing the posterior of class 1 with a threshold τ:

$$\hat{\mathbf{y}} = \mathbb{1}\{\sigma_1(x) \geq \tau\} \tag{5}$$

We calibrate a single threshold τ^* per ontology in its development split. This split is constructed by holding out 10% of the pairs / triples from the training set, which are reserved only for threshold calibration and are not used in training:

$$\tau^* = \arg \max_{\tau \in \{0.05, 0.10, \ldots, 0.95\}} \text{F1-score}\big(\hat{\mathbf{y}}_{\text{dev}}, y_{\text{dev}}\big), \tag{6}$$

Here, $\hat{\mathbf{y}}_{\text{dev}}$ is a binary vector of model predictions and y_{dev} is a binary vector of ground-truth labels on the development set.

Negative sampling. The training files contain only positive instances, yet a binary classifier also needs negative samples. Using all unseen combinations as negatives is infeasible since an exhaustive crawl would require scoring $O(|\mathcal{T}|^2)$ child–parent pairs for subtask C and $O(|\mathcal{T}|^2|\mathcal{R}|)$ head–relation–tail triples for subtask D. For example, in the SWEET ontology, fewer than 0.0002% of all possible pairs actually hold the taxonomic relation, yielding a large and imbalanced dataset. Instead, for each original training sample, we generate a small, fixed batch of k synthetic negatives using the task-specific rules described below:

- **Task C** $k{=}9$: keep c and replace p with a random non-ancestor p'.
- **Task D** $k{=}19$: keep h and corrupt exactly one slot (h, r', t), or (h, r, t').

In principle, the optimal number of negative samples per positive k could vary depending on the ontology domain and the presence or absence of definition lines in the prompt. A detailed search for the optimal value per task per ontology was not feasible within our time budget, so we fixed them once for the entire study: nine negative samples for subtask C and 19 negative samples for subtask D. These values strike a practical balance between class balance and training time and are applied unchanged to all three ontologies and to every ablation setting.

Loss and optimisation. A class-weighted cross-entropy

$$\mathcal{L}_{\text{CE}} = -\tfrac{1}{B}\sum_b (w_1 y_b \log \sigma_{b,1} + w_0(1 - y_b) \log \sigma_{b,0}) \tag{7}$$

where B is the batch size, $y_b \in \{0, 1\}$ is the ground-truth label for the b-th instance (1 for positive, 0 for negative), and $\sigma_{b,1}$, $\sigma_{b,0}$ are the predicted probabilities for the positive and negative classes, respectively. The class weights are set as $w_1 = k$ and $w_0 = 1$, where k controls the penalty applied to false negatives. We keep the hyper-parameters from Subtask B: AdamW optimiser, learning rate 2×10^{-5}, batch size 16, max length 512 if definitions are enabled, otherwise 128 and 10 epochs.

Inference. Let $\mathcal{T}_{\text{test}}$ be the set of ontology types that occur anywhere in the test split and let $|\mathcal{T}_{\text{test}}| = n$.

- *Taxonomy Discovery.* We enumerate every ordered child–parent pair (c, p) with $c, p \in \mathcal{T}_{\text{test}}$ and $c \neq p$ ($n(n-1)$ pairs in total). Each pair is placed in the template shown above and passed through the encoder, resulting in the posterior of the positive class $\sigma_1(c,p)$. The link *isChildOf*(c,p) is accepted if $\sigma_1(c,p) \geq \tau^*$. Because decisions are independent, a child can receive multiple parents, mirroring the polyhierarchy allowed in the training data.

- *Non-Taxonomic Relation Extraction.* Let $\mathcal{R}_{\text{test}}$ be the set of relations to evaluate. For every ordered type pair (h,t) with $h,t \in \mathcal{T}_{\text{test}}$, $h \neq t$, and for every $r \in \mathcal{R}_{\text{test}}$, we form a candidate triple (h,r,t) that is also cast into template shown above and processed through encoder; the search space is $n(n-1)|\mathcal{R}_{\text{test}}|$. The relation $r(h,t)$ is predicted to be true when $\sigma_1(h,r,t) \geq \tau^*$.

Using definitions (+Def regime). When we use the type definition enrichment technique, we first retrieve a one-sentence glossary for every type using the pipeline of Section 3.1.3 and then append that glossary to the corresponding `[CHILD]`, `[PARENT]`, `[HEAD]` or `[TAIL]` line in the input template. All other components, that is, the DeBERTa encoder, the negative sampling scheme, the class weighted loss, and the calibrated decision threshold,remain exactly as in the vanilla configuration. Only the text presented to the model will be changed.

3.2.2 Similarity-based Candidate Filtering

Random negative sampling produces arbitrary pairs or triples, allowing the model to easily separate positives. To create harder negatives and reduce the test time search space, we prune the set of all available types to the K most *semantically* similar types, based on the assumption that similar types are more likely to be linked. Every type is embedded with *all-MiniLM-L6-v2* Sentence-BERT model[8], yielding a 384-dimensional vector, and the resulting vectors are indexed in FAISS.[9] For a given type t we define

$$\mathcal{N}_K(t) = \{\, t^{(1)}, \ldots, t^{(K)} \}, \tag{8}$$

its K nearest neighbours in this embedding space, computed within the current split so that training never sees test embeddings and vice versa. We fix $K = 100$ for all experiments. This is a pragmatic compromise given our time constraints, although a search for an optimal K could improve the results. When enabled, the $\mathcal{N}_K(t)$ types modify two steps of the baseline solutions:

- *Negative sampling*: corrupt pairs/triples only with elements from $\mathcal{N}_K(t)$.
- *Inference*: compare a child (or head/tail) only to its K neighbours, reducing the search to $O(K\,|\mathcal{T}|)$.

The rest of the baseline pipeline is intact.

4. Data availability Statement

All raw task datasets (training and test splits for OBI, MatOnto, and SWEET) are released by the organisers of the LLMs4OL Challenge 2025. External resources that we used for augmentation and definition mining, i.e. Wikipedia and Wiktionary REST endpoints, USGS/NOAA/ADS glossaries, MeSH, PubChem, and UniProt APIs, are open-access services that can be queried without restriction.

We publish every derivative artefact generated (i.e., augmented training sets for term typing, mined type definitions, and all configuration / training scripts) in a dedicated

[8] `all-MiniLM-L6-v2`
[9] `FAISS`

GitHub repository, mentioned in the abstract, at the time of publication, enabling full reproduction of the reported results.

5. Experiments and Results

All results are evaluated with the official LLMs4OL evaluation script provided by the organisers. For every subtask, the script returns micro-averaged precision, recall and F1-score, i.e. the counts of true/false positives and negatives are aggregated globally over the whole test file before the three metrics are calculated. All results of our experiments are collected from the challenge leaderboard platform and summarised in Tables 3–5. No alternative scoring or post-processing is applied on our side.

5.1 Term typing (Subtask B) — RQ1

Table 3. *Sub-task B: micro-averaged scores on the official test splits.* ***Highest*** *value per ontology and metric is* ***bolded****. Settings:* ***V*** *- Vanilla baseline (no A, no D),* ***A*** *- using data augmentation,* ***D*** *- using term definitions,* ***A+D*** *- both techniques.*

Ontology	Setting	F1-score	Precision	Recall
OBI	V	0.491	0.363	0.759
	A	**0.839**	0.788	**0.897**
	D	0.528	0.390	0.816
	A+D	0.828	**0.828**	0.828
MATONTO	V	0.097	0.065	0.189
	A	0.500	0.465	0.541
	D	0.312	0.300	0.324
	A+D	**0.667**	**0.614**	**0.730**
SWEET	V	0.543	0.463	0.657
	A	0.557	0.505	0.621
	D	0.645	**0.640**	0.650
	A+D	**0.653**	0.588	**0.735**

Across the three ontologies, the weakest scores are for the vanilla setting (**V**). This is expected due to the limited data available for training. In OBI and MatOnto, the largest single gain comes from data augmentation (**A**), whereas on SWEET, adding mined term definitions (**D**) has a greater impact. Combining the two signals (**A+D**) usually helps surpass the individual variants on MatOnto and SWEET, and only falls slightly below using augmented data on OBI.

In summary, each enrichment outperforms the vanilla setting, and in two of the three ontologies, the joint use of augmentation plus definitions yields the best overall performance, giving an answer to **RQ1** that these enrichment techniques could substantially boost performance on term typing.

5.2 Taxonomy discovery (Subtask C) — RQ2

Similarity-based filtering is the dominant factor in taxonomy discovery. Replacing random corruptions with neighbour-based negatives (**F**) lifts the micro-F1-score from near zero to practical levels in the three ontologies: +0.33 in OBI, +0.43 in MatOnto and +0.18 in SWEET. The gain is more obvious if we measure the difference in recall (e.g. 0.71 vs. 0.01 on MatOnto), indicating that pruning the search to semantically plausible parents helps the model recover true edges that would otherwise be swamped by noise.

Table 4. *Subtask C: micro-averaged scores on the official test splits.* ***R****: random negatives;* ***F****: similarity filter;* ***D****: type definitions;* ***F+D****: both filter and definitions. Best value per ontology/metric in* ***bold****.*

Ontology	Setting	F1-score	Precision	Recall
OBI	R	0.006	0.056	0.003
	F	0.333	0.240	**0.544**
	D	0.006	0.065	0.003
	F+D	**0.397**	**0.315**	0.537
MatOnto	R	0.020	0.098	0.011
	F	**0.447**	**0.327**	0.709
	D	0.025	0.106	0.014
	F+D	0.396	0.269	**0.745**
SWEET	R	0.002	0.014	0.001
	F	0.182	0.139	**0.263**
	D	0.000	0.000	0.000
	F+D	**0.252**	**0.289**	0.224

Type definitions (**D**) offer a little gain on their own when the list of candidates is still dominated by random negatives. However, once filtering is applied, the definitions increase the results further. The (**F+T**) setting achieves the best overall F1-score in OBI (0.40) and SWEET (0.25), mainly through precision gains of 7-15 pp, while in MatOnto it trades a small drop in precision for still higher recall. Thus, semantic filtering is essential for taxonomy discovery, and type-definition glossaries act as a fine-grained booster once the search space has been denoised, jointly answering **RQ2**.

5.3 Non-taxonomic relation extraction (Subtask D) — RQ2

Table 5. *Sub-task D (SWEET).* ***R****: random negatives;* ***F****: similarity filter;* ***D****: type definitions;* ***F+D****: both filter and definitions. Highest value per metric in* ***bold****.*

Setting	F1-score	Precision	Recall
R	0.058	0.051	0.066
F	0.391	0.260	**0.788**
D	0.000	0.000	0.000
F+D	**0.532**	**0.426**	0.709

For subtask D only the SWEET ontology was evaluated. The pattern observed in subtask C is also observed in subtask D for SWEET. Filtering the candidate space (**F**) is indispensable, lifting micro-F1-score from 0.06 to 0.39 and, notably, pushing the recall to 0.79. Type definitions (**D**) alone cannot overcome the noise in a random candidate list, but when they are added on top of the filter (**F+D**) they deliver a further jump to the best overall score ($F_1 = 0.53$, precision 0.43). Thus, semantic filtering is the foundation; definitions provide an extra boost once the search space has been cleaned. This finding on non-taxonomic relation extraction additionally contributes to **RQ2**.

6. Conclusion

We showed that three simple, ontology-agnostic data–layer heuristics, i.e., data augmentation for term typing, automatic definition mining for terms and types, and semantic-based candidate filtering for taxonomy discovery and non-taxonomic relation extraction, substantially improve a single DeBERTa-v3 model across all LLMs4OL subtasks. For term typing, both data augmentation and term definition mining outperform the vanilla setting, and their combination gives the best F1 score on two out of three ontologies.

For taxonomy discovery and non-taxonomic relation extraction, candidate filtering is essential, lifting F1-score from almost zero to 0.18-0.45; adding type definitions on these cleaner candidate lists yields further gains, confirming that definitions help once noise is reduced.

In general, the three data enrichment techniques attack different bottlenecks: rare-class sparsity, context deficit, and quadric explosion, and thus combine well without changing model architecture or hyperparameters.

A more fine-grained evaluation requires a systematic ablation study of our data enrichment toolbox. Future work should therefore (i) examine how individual components: different synonym sources or definition APIs—affect performance; (ii) tune the key hyperparameters per ontology, in particular the neighbour-list length K for candidate filtering and the negative-to-positive ratio k; (iii) refine the augmentation and definition pipelines with deeper domain analysis for each ontology.

Competing interests

The authors declare that they have no competing interests.

Author contributions

Insan-Aleksandr Latipov: Conceptualization, Data Curation, Software, Formal Analysis, Investigation, Methodology, Writing - Original Draft.

Mike Holenderski: Supervision, Funding acquisition, Writing - Review & Editing

Nirvana Meratnia: Supervision, Writing - Review & Editing

Funding

This work was supported by the AIMS5.0 project under grant agreement no. 101112089.

References

[1] H. Babaei Giglou, J. D'Souza, and S. Auer, "Llms4ol: Large language models for ontology learning", in *International Semantic Web Conference*, Springer, 2023, pp. 408–427.
[2] S. Deng, Y. Ma, N. Zhang, Y. Cao, and B. Hooi, "Information extraction in low-resource scenarios: Survey and perspective", in *2024 IEEE International Conference on Knowledge Graph (ICKG)*, IEEE, 2024, pp. 33–49.
[3] O. Perera and J. Liu, "Exploring large language models for ontology learning", 2024.
[4] H. B. Giglou, J. D'Souza, and S. Auer, "Llms4ol 2024 overview: The 1st large language models for ontology learning challenge", *arXiv preprint arXiv:2409.10146*, 2024.
[5] H. Babaei Giglou, J. D'Souza, N. Mihindukulasooriya, and S. Auer, "Llms4ol 2025 overview: The 2nd large language models for ontology learning challenge", *Open Conference Proceedings*, 2025.
[6] A. Bandrowski et al., "The ontology for biomedical investigations", *PloS one*, vol. 11, no. 4, e0154556, 2016.

[7] K. Cheung, J. Drennan, and J. Hunter, "Towards an ontology for data-driven discovery of new materials.", in *AAAI Spring Symposium: Semantic Scientific Knowledge Integration*, 2008, pp. 9–14.

[8] R. Raskin and M. Pan, "Semantic web for earth and environmental terminology (sweet)", in *Proc. of the Workshop on Semantic Web Technologies for Searching and Retrieving Scientific Data*, vol. 25, 2003.

[9] P. Cimiano and J. Völker, "Text2onto: A framework for ontology learning and data-driven change discovery", in *International conference on application of natural language to information systems*, Springer, 2005, pp. 227–238.

[10] M. N. Asim, M. Wasim, M. U. G. Khan, W. Mahmood, and H. M. Abbasi, "A survey of ontology learning techniques and applications", *Database*, vol. 2018, bay101, 2018.

[11] R. Du, H. An, K. Wang, and W. Liu, "A short review for ontology learning: Stride to large language models trend", *arXiv preprint arXiv:2404.14991*, 2024.

[12] A. S. Lippolis et al., "Ontology generation using large language models", in *European Semantic Web Conference*, Springer, 2025, pp. 321–341.

[13] Y. Lu et al., "Unified structure generation for universal information extraction", *arXiv preprint arXiv:2203.12277*, 2022.

[14] X. Wang et al., "Instructuie: Multi-task instruction tuning for unified information extraction", *arXiv preprint arXiv:2304.08085*, 2023.

[15] J. Xu, M. Sun, Z. Zhang, and J. Zhou, "Chatuie: Exploring chat-based unified information extraction using large language models", *arXiv preprint arXiv:2403.05132*, 2024.

[16] P. He, X. Liu, J. Gao, and W. Chen, "Deberta: Decoding-enhanced bert with disentangled attention", *arXiv preprint arXiv:2006.03654*, 2020.

[17] A. Hurst et al., "Gpt-4o system card", *arXiv preprint arXiv:2410.21276*, 2024.