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.2897

Published: 01 Oct. 2025

ELLMO at LLMs4OL 2025 Tasks A and D: LLM-Based Term, Type, and Relationship Extraction

Ryan Roche[1], Ryn Gray[1,*], Jaimie Murdock [1], and Douglas C. Crowder[1]

[1]Sandia National Laboratories, Albuquerque NM

*Correspondence: kgray1@sandia.gov

Abstract. This paper presents an approach to building ontologies using Large Language Models (LLMs), addressing the need in many domains for quality knowledge data extraction from vast stores of text data. In particular, we focus on extracting terms and types from text and discovering relationships between types. This work was completed as part of the 2025 LLMs4OL Challenge, where quality training and testing data, as well as several defined tasks were provided. Many teams competed to produce the best output data across many domains. Our methodology involved prompt engineering, classification, clustering, and vector databases. For the first task, discovering terms and types, we used two methods, (1) directly tailoring prompts to find the terms and types separately and (2) an approach that discovered terms and types simultaneously and then classified them afterwards. For discovering relationships, we used clustering and vector databases to attempt to reduce the number of potential edges; then, we queried the LLM for probabilities for each of the potential edges. While our findings indicate promising results, further work is necessary to address challenges related to processing large datasets, particularly in optimizing efficiency and accuracy.

Keywords: Term Extraction, Type Extraction, Relationship Discovery, LLMs4OL Challenge, Ontology Construction

1. Introduction

As the amount of scientific knowledge continues to grow, it is becoming increasingly important to organize information into formats that allow scientists to understand connections between concepts. Large language models (LLMs) such as ChatGPT [1], [2], Llama [3], and Gemma [4], have recently gained substantial attention for their abilities to analyze large quantities of information and help scientists efficiently find relevant information. In fact, more recent developments have allowed LLMs to perform logical reasoning [5], which opens up new opportunities for using LLMs as research assistants and co-pilots [6].

Unfortunately, LLMs are also known for creating false information - a phenomenon known as “hallucination.” These hallucinations make it difficult to use LLMs, alone, for making complex and expensive decisions, such as designing new experiments. At present, these hallucinations force researchers to use highly-manual processes where

pieces of critical information must be individually verified. To enable more automatic LLM workflows, hallucinations must be mitigated automatically. Several approaches to hallucination mitigation have been proposed, including LLM finetuning [7] and retrieval augmented generation (RAG) [8], [9] workflows.

Additionally, symbolic knowledge representations, such as knowledge graphs and ontologies, can be used to not only mitigate hallucinations but also help LLMs efficiently retrieve information that is relevant to a given query [10], [11]. While there are many different types of symbolic knowledge representations, we believe that ontologies are particularly well suited for scientific knowledge synthesis, given that they can serve as templates and enable logical reasoning. By serving as templates, ontologies can help us automatically detect missing information. And, by enabling logical reasoning, ontologies can enable automatic fact checking.

Despite the promise of ontologies, the process of creating ontologies has relied on manual or semi-automated [12] workflows that will not be appropriate for internet-scale data. Consequently, while ontologies are widely used within specific communities, their adoption by the wider scientific community has remained relatively small. To enable efficient generation of internet-scale ontologies, or collections of ontologies, we need to develop fully-automated workflows. Because of their broad knowledge and their natural language processing (NLP) abilities, LLMs are prime candidates for ontology learning. Unfortunately, naive prompting of LLMS for ontology learning (OL) results in mediocre performance [13]. This work uses the LLM4OL challenge datasets [14] to explore methods for enabling LLM-powered OL. Specifically, we explore how LLMs can be used to extract terms and types (LLM4OL Task A) and define non-taxonomic relationships (LLM4OL Task D).

2. Related Work

There is a large body of literature exploring how to combine LLMs with symbolic knowledge representations. Much of the literature focuses on combining LLMs with knowledge graphs, either by using LLMs to build knowledge graphs, using knowledge graphs to inform LLMs, or some combination thereof. See [15], [16], [17], [18], [19] for reviews on this topic.

Many recent works have also explored how LLMs can be combined with ontologies [20], [21]. [22] demonstrated that LLMS could be used to verify ontologies. [23] considered how competency questions could be used to engineer ontologies. [24] demonstrated that LLMs and ontological reasoning could be combined to elucidate deep relationships in data. [25] demonstrated LLMs to align existing ontologies. [26] used LLMs to augment ontologies. [27] demonstreated that LLMs could be used to automatically build ontologies, but noted challenges in creating ontological properties. [28] employed ontology reuse and prompt engineering for ontology learning. [13] provided strategies for evaluating LLM-generated ontologies.

The most pertinent body of related work comes from previous instantiations of the LLM4OL challenge. [29] notes that zero-shot learning does not work for ontology learning. [30] provides results from the most recent competition. Compared to previous competitions, this year's competition contained different datasets, including adding Task A (extracting terms and types). Because Task A was added this year, there are no results from previous year to compare to.

Task D from this year's competition is equivalent to Task C from last year's competition. Two teams worked on Task C last year, silp_nlp and the Phoenixes. The first

team [31], silp_nlp, prompted the LLM with all potential types and relations and asked it to return all relationships. The second team [32], the Phoenixes, found which edges were likely to exist by first finding the cosine distance between nodes and then setting a threshold to determine which nodes were close enough to have an edge between them. Then, they queried an LLM to determine whether a relationship actually existed between those nodes.

We expand on these concepts by first using an LLM to cluster the terms to find types likely to have connections and then querying the LLM for a probability for each of these potential connections.

3. Methods

3.1 Task A

3.1.1 Task Overview

Task A - Text2Onto of the LLMs4OL 2025 Challenge aims to bridge the gap between unstructured text and structured ontological knowledge through the extraction of domain-relevant terminologies and their associated types [33]. This task is divided into two subtasks: **SubTask A1 (Term Extraction)**, which involves identifying key domain-specific terms that serve as potential ontology instances, and **SubTask A2 (Type Extraction)**, which involves extracting the abstract categories or ontology classes to which the term belongs. The overarching goal is to facilitate the automated construction of ontologies from raw text, thereby supporting several downstream tasks such as semantic search, reasoning, and knowledge integration.

3.1.2 Datasets

The organizers of the challenge provided three independent datasets that were specific to a range of domains. These datasets were *Ecology*, *Scholarly*, and *Engineering* as shown in Table 1. For Task A, we concentrated on the *Scholarly* and *Engineering* datasets due to their balanced distributions of documents and associated terms/types, which provided a suitable foundation for evaluation.

3.1.3 Approach Overview

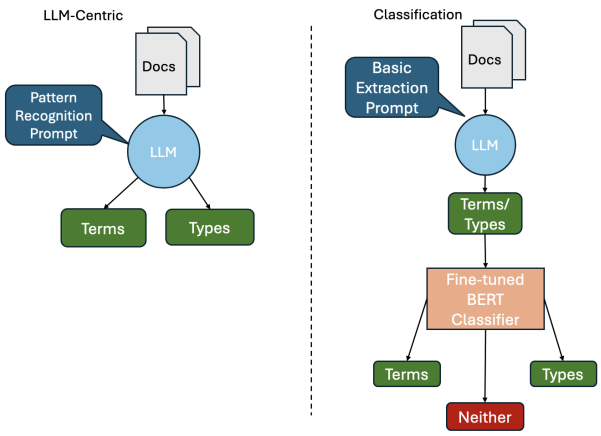


Figure 1. *Overview of our methodologies for Task A.*

We investigated several procedures for extracting terms and types (as shown in Figure 1), two of which demonstrated the highest accuracy when evaluated against the training datasets. The first (LLM-Centric) procedure employed a LLM, leveraging a predefined pattern recognition prompt. In this approach, terms and types were separately extracted through distinct LLM prompts. The second (Classification) procedure utilized a unified, comprehensive prompt to extract both terms and types simultaneously in a single LLM invocation. Following the extraction phase in the Classification procedure, a fine-tuned BERT classification model was employed to categorize each phrase as either a *term* or a *type*. For both procedures, we used the `Mistral-Small-3.1` LLM due to its high performance and relatively small number of model parameters [34].

Table 1. *Characteristics of the provided datasets for Task A*

	Train Terms	Train Types	Train Docs	Test Docs
Ecology	44	5734	2000	482
Scholarly	246	260	40	10
Engineering	1143	772	83	21

3.1.4 LLM-Centric Procedure

The LLM-Centric procedure was designed to be generalizable across various domains, represented by the various Task A datasets, while maintaining acceptable performance. The operation leverages prompt tuning to encouraging the LLM to recognize patterns within provided documents.

Terms

To extract the terms, the prompt was constructed using *hard prompting*, a method involving the manual creation of tailored prompts [35], [36]. Although this approach can be labor-intensive during prompt tuning, it proved sufficient for guiding the model toward recognizing a generalizable syntactic pattern. In particular, the LLM was directed to identify instances of *syndetic coordination* which are sentences that present a list of terms followed by a coordinating conjunction [37]. The model was instructed to extract all terms appearing within such constructions with the following prompt:

TASK: Extract all terms that are part of syndetic coordination from the given text.
DEFINITION: Syndetic coordination is a coordinate construction where two or more syntactically equivalent elements are linked using coordinating conjunctions ("and," "or," "but," "nor," "yet," "so," or "for").

EXAMPLE:
Input: "The students studied biology, chemistry, physics, and mathematics during their final semester."
Output:
biology
chemistry
physics
mathematics

INSTRUCTIONS:
1. Identify all syndetic coordination patterns in the text
2. Extract only the coordinated terms (not the conjunctions)

TEXT:
{text}

FORMAT: Only output one term per line without any other formatting. Ensure all terms are in the singular form.

This prompt design was informed through manual inspection of the target documents. Notably, the *Scholarly* subset contained a high frequency of syndetic coordination structures, making this approach especially effective for extracting accurate terms from such sources.

Types

The types were extracted using a simple prompt that instructed the model to identify and extract types based on its own understanding. Alternative prompting strategies, such as few-shot examples [38] or the inclusion of explicit definitions, yielded worse results on the provided training data. The model was instructed using the following prompt:

Extract all the types (classes) from the following text:
{text}

FORMAT: Only output one term per line without any other formatting. Ensure all terms are in the singular form.

3.1.5 Classification Procedure

This Classification procedure employed LLM prompting followed by a fine-tuned LLM classifier trained on the provided data. Unlike the LLM-Centric procedure, the LLM was prompted to extract both terms and types simultaneously using a unified prompt. This prompt was created by merging the individual prompts previously used in the LLM-Centric approach as follows:

```
TASK: Extract all terms (entities) AND all types (classes) from the given text.

TEXT:
{text}

FORMAT: Output only one term per line with no formatting, bullets, quotes, or
additional text.
```

Each domain had a subset of rules applied to this base prompt in order for the model to extract patterns specific to that dataset. The following *Scholarly* subset of rules were applied to the base prompt:

```
- **Case System Recognition**: For case names (dative, genitive, accusative,
etc.), include "case" in the term (e.g., "dative case" not just "dative")
- **Voice Completion**: For voice terms (active, passive, middle), include "voice"
in the term (e.g., "active voice" not just "active")
- **Number Specification**: For number/quantity terms that are incomplete,
include the full specification (e.g., "other number" not just "other")
```

The following *Engineering* subset of rules were applied to the base prompt:

```
- Extract ALL complete noun phrases and technical terms
- Extract ALL compound terms with hyphens, spaces, or parentheses
- Extract ALL units of measurement (with or without prefixes)
- Extract ALL scientific abbreviations and acronyms
- Extract ALL numerical expressions with units
- Extract ALL temperature scales and specialized measurements
- Extract ALL proper nouns and technical names
- Extract both singular and plural forms when they appear
- Do NOT extract partial words, sentence fragments, or incomplete phrases
- Do NOT extract generic words like "various", "different", "base", etc.
- Focus on domain-specific technical terminology
```

Following the extraction phase, a fine-tuned classification model, `bert-base-uncased`, was employed to label each element as either a *term* or a *type*. The fine-tuning dataset was divided into training (80%) and test (20%) sets and stratified, maintaining the original class proportions. AdamW was utilized as the optimizer with a learning rate of 2×10^{-5}. Early stopping with patience of 3 epochs was used to prevent overfitting.

This approach resulted in notable performance improvements in domains characterized by significant overlap between terms and types. The key advantage of this method lies in offloading the disambiguation task from the pretrained, but not fine-tuned, LLM. Instead of requiring the LLM to extract the same element twice, once as a term and once as a type, the classification model was tuned to classify each entity. We decided to test 2 different strategies for fine-tuning the classification model. The first strategy included two categories: *terms* and *types*. The second strategy included an additional class labeled *neither*, which allowed the classification model to exclude some of the incorrectly predicted terms or types, thereby increasing the precision. The data for training this class was curated by collecting the incorrectly predicted terms and types from the LLM-Centric procedure.

Depending on the domain, an additional class was added to the tuning of the model titled *both*. This special class was only necessary for domains with overlapping ground-truth terms and types. This strategy proved particularly effective for the *Scholarly* dataset, where such overlaps frequently occurred. In contrast, the *Engineering*

domain exhibited only a single overlapping instance, despite having considerably more documents. Consequently, for this domain, the classification model was fine-tuned without the *both* category.

3.2 Task D

3.2.1 Task Overview

Task D - Non-Taxonomic Relation Extraction of the LLMs4OL 2025 Challenge aims to build non-taxonomic relations between types. This task is divided into three subtasks, which are also different datasets: **SubTask D1 - SWEET** (an environmental and geoscience dataset), **SubTask D2 - FoodOn** (a food ontology dataset), and **SubTask D3 - GO** (a genomic dataset). The goal for each of these datasets is, given lists of the types and potential relationships, can we match the correct types and relationships. For example, "erode, equivalentClass, erosion" is found using the types "erode" and "erosion" and the relationship "equivalentClass". Many of the edges are directional. For example, when connecting types with "term replaced by", "A term replaced by B" gives a different meaning that "B term replaced by A". Some edges are symmetric, such as "equivalentClass". Most types in the datasets are only connected with one or two other terms, creating fairly sparse datasets. Table 2 shows the 3 Task D datasets and how they increase in terms of potential types and number of edges.

Table 2. *The number of types and relationships given for each Task D dataset as well as the number of edges that exist (train answers) for each dataset.*

	Train Types	Train Relationships	Train Answers	Test Types	Test Relationships
SWEET	662	3	360	297	2
FoodOn	2838	6	1450	1233	4
GO	9622	6	11606	5189	6

3.2.2 Approach Overview

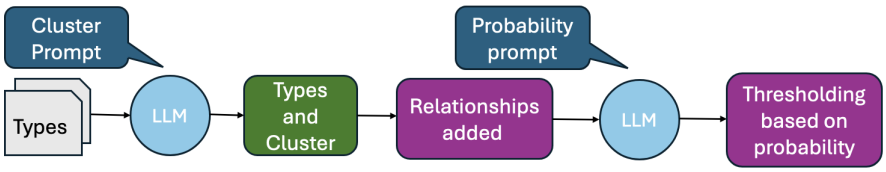


Figure 2. *Overview of our methodology for Task D*

Our approach involved first, reducing the number of potential edges, and second, determining a probability for whether the edge existed based on LLM output, as seen in Fig: 2. We reduce the number of potential edges with two of methods, either by using an LLM to cluster the types (Clustering procedure) or by using a vector database to find the nearest n types (Vector Database procedure). Once similar types were identified, we assumed that all edges existed between all similar nodes. We could then use an LLM to give a probability that a specific edge existed for each of these pairs.

3.2.3 Clustering Procedure

To cluster types, we used `llama-3.2-90B-Vision-Instruct` and the following prompt:

> Cluster all of these types. Give what cluster every type would be in. Format it as "term, cluster" in csv format. Types: {List of all types}

This gave each node a grouping. Once we had this grouping, we assumed that every edge existed between all pairs of nodes in the grouping. This format worked best for the *SWEET* dataset, as we could run the next step (finding probabilities of each edge) within a reasonable time. For the *FoodOn* dataset, we batched the clustering by giving 500 types at a time, and then extracted the names of the clusters and passed those in for the next cluster. In the data below, we prompted the LLM to create 10 clusters during the initial step (500 nodes), and included the line, "you may add a few clusters if necessary" in the prompt for the rest of the batches.

Note that the clustering method reduces the number of potential edges by $1/c$, where c is the number of clusters. The results in this paper are included from prompting for 10 clusters and including the additions in the next prompts. We found that, without this, we got very low scores and the cluster names were highly variable. Even with this prompt, the LLM sometimes produced clusters with the descriptor "None" for some of the more difficult types.

3.2.4 Vector Database Procedure

Another attempt to reduce the number of potential edges involved vector databases. We used `bert-base-uncased` to create the vector database and then queried the vector database to find the n nearest neighbors, where $n = 35$ for the *SWEET* dataset and $n = 500$ for the *FoodOn* dataset.

3.2.5 Probabilities for Edges

Prompting without Examples We asked the LLM to output probabilities that edges would exist by using the following prompt:

> Please provide a list in CSV format how likely each of these connections are. The potential connections are: connections. Give ONLY the csv format back as Node1, EdgeType, Node2, probability. Use no extra words and make sure it is in valid csv form.

Prompting with Examples We tested if we could improve the edge probability estimates by using few-shot learning by adding several examples to the prompt from the **Prompting without Examples** section. Generally, we used one example from each potential relationship, although we did not include any relationship that had only one edge in the example data. Additionally, we included one example that was not an edge and gave it a low probability.

4. Results

4.1 Task A

Table 3. *LLM-Centric training dataset results for Task A using Mistral-Small-3.1*

	F1 Score	Precision	Recall
Scholarly	0.7940	0.6752	0.9634
Engineering	0.4947	0.4504	0.5486

Terms

	F1 Score	Precision	Recall
Scholarly	0.6790	0.5397	0.9154
Engineering	0.3783	0.2958	0.5246

Types

Table 4. *LLM-Centric testing dataset results for Task A using Mistral-Small-3.1*

	F1 Score	Precision	Recall
Scholarly	0.3652	0.2530	0.6562
Engineering	0.4360	0.5656	0.3547

Terms

	F1 Score	Precision	Recall
Scholarly	0.5524	0.3867	0.9667
Engineering	0.0952	0.0526	0.5000

Types

Table 5. *LLM-Centric testing dataset results for Task A using Claude Sonnet 4*

	F1 Score	Precision	Recall
Scholarly	0.2486	0.1503	0.7188
Engineering	0.6072	0.8464	0.4735

Terms

	F1 Score	Precision	Recall
Scholarly	0.3951	0.3137	0.5333
Engineering	0.6750	0.6136	0.75

Types

Table 6. *Classification training dataset results without the **neither** class for Task A using Mistral-Small-3.1*

	F1 Score	Precision	Recall
Scholarly	0.7417	0.6078	0.9512
Engineering	0.5925	0.6815	0.5241

Terms

	F1 Score	Precision	Recall
Scholarly	0.8256	0.7492	0.9192
Engineering	0.4642	0.4332	0.5000

Types

Table 7. *Classification training dataset results with the **neither** class for Task A using Mistral-Small-3.1*

	F1 Score	Precision	Recall
Scholarly	0.8553	0.8831	0.8293
Engineering	0.6195	0.7824	0.5127

Terms

	F1 Score	Precision	Recall
Scholarly	0.8623	0.8963	0.8308
Engineering	0.5158	0.6626	0.4223

Types

Table 8. *Classification test dataset results without the **neither** class for Task A using Mistral-Small-3.1*

	F1 Score	Precision	Recall
Scholarly	0.2517	0.1622	0.5625
Engineering	0.5011	0.6501	0.4077

Terms

	F1 Score	Precision	Recall
Scholarly	0.4242	0.3043	0.7000
Engineering	0.1461	0.0816	0.6944

Types

Table 9. *Classification test dataset results with* ***neither*** *class for Task A using Mistral-Small-3.1*

	F1 Score	Precision	Recall
Scholarly	0.2857	0.3333	0.2500
Engineering	0.5588	0.8118	0.4260

Terms

	F1 Score	Precision	Recall
Scholarly	0.5098	0.6190	0.4333
Engineering	0.2051	0.1481	0.3333

Types

We noticed that the size of the model can have a large impact on the results (compare Table 4 to Table 5). With `Claude Sonnet 4`, each metric in the *Engineering* results was significantly greater than those generated with the smaller-model (`Mistral-Small-3.1`). On the flip side, `Claude Sonnet 4` produced lower scores on the *Scholarly* test dataset.

For the *Scholarly* dataset, the LLM-Centric procedure outperformed the Classification procedure on the test data, as measured using F1 and recall scores. However, the Classification procedure performed better in terms of precision on the test data. Full results are shown in Tables 4 and 9. For the *Engineering* dataset, the Classification procedure outperformed the LLM-Centric procedure on the test data, as measured using F1 and recall scores. Again, results are shown in Tables 4 and 9.

We observed that, by including the *neither* class in the Classification procedure, classification results were improved, as measured using F1 scores (compare Table 8 to Table 9 and Table 6 to Table 7). By including the *neither* class, precision was generally improved, and recall was generally decreased, as would be expected.

There was a notable decrease in performance between OL performance on the training and testing datasets (compare Table 3 to Table 4 and Table 7 to Table 9).

4.2 Task D

4.3 Clustering Procedure

The output from just the clustering step can be seen in Table: 10. The purpose of this step is to keep recall high by finding similar nodes that will undergo a future LLM-based filtering step. Without undergoing the LLM0based filtering step, we expect precision to be low. Note that including every possible edge will give a recall score of 1, so the goal of this step is to keep the recall score high. We can see that the clustering procedure works well for the *SWEET* dataset (recall remains high), but clustering results in a low recall for the *GO* dataset. Recall for the *FoodOn* dataset is between recall for the *SWEET* and *GO* datasets.

Table 10. *Performance for each dataset from the clustering step for Task D. After clustering, we expect relatively high recall but low precision.*

Dataset	F1 Score	Precision	Recall
SWEET	0.0024	0.0048	0.7306
FoodOn	0.00 01	0.0002	0.3169
GO	0.0001	0.0002	0.0975

4.4 Vector Database Procedure

When using the vector database procedure, we performed some initial studies to determine the number of nearest neighbors that we should use. As shown in Table 11, for the *SWEET* dataset, we elected to use 35 neighbors, and for the *FoodOn* dataset,

we elected to use 500 neighbors. As in Table 10, this step resulted in a high recall, but a relatively low precision, because additional filtering needed to be performed by the LLM. Compared to clustering, vector database methods resulted in lower recall and comparable precision, and the methods were not scalable to larger datasets (i.e., *GO*).

Table 11. *Performance for each dataset after just the vector database step.*

Dataset	F1 Score	Precision	Recall	Number of Neighbors
SWEET	0.0040	0.0080	0.4770	35
FoodOn	0.0001	0	0.3149	500

4.5 Probability Procedure

LLMs were asked to produce probabilities that there were edges between pairs of nodes, downselected using clustering or vector database methods. We then thresholded these probabilities to determine which edges should be assigned to nodes. As shown in Figure 3, recall, precision, and F1 were all affected by the choice of threshold. Generally, thresholds less than 0.8 resulted in near-zero precision and F1 scores. By setting the threshold at 1.0, we maximized precision and F1, although recall suffered.

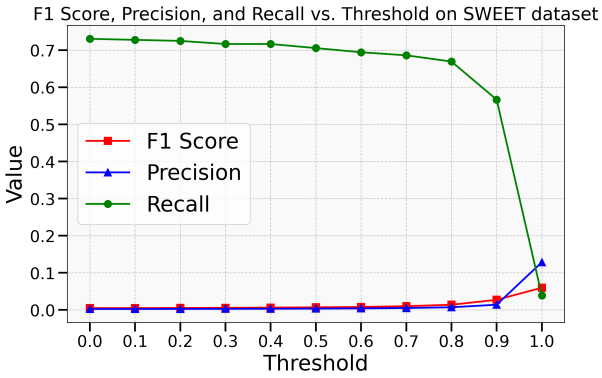


Figure 3. *Effects of different thresholds for LLM-produced probabilities of edges for the SWEET dataset (Task D).*

Figure 4, shows distribution of the edge probabilities estimated by the LLMs. When the prompt included examples, the LLMs produced more edge probabilities greater than or equal to 80%, possibly because the provided examples included probabilities that were closer to 1. Because more edges were assigned higher probabilities of existence, the precision and F1 scores were negatively impacted by including examples, as shown in Tables 12 and 13.

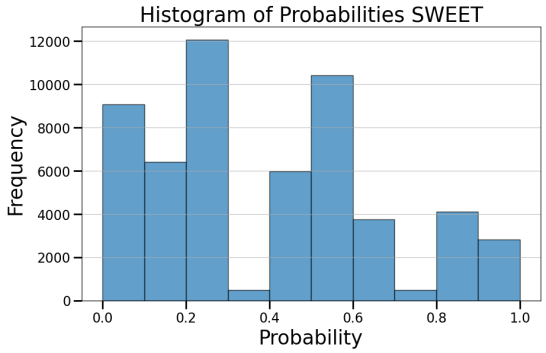
Table 12. *This table shows the difference in the output with the basic and example on the training dataset.*

Dataset	F1 Score	Precision	Recall	Prompt Type
SWEET	0.0528	0.0277	0.5662	Without Examples
SWEET	0.0481	0.0251	0.5941	With Examples

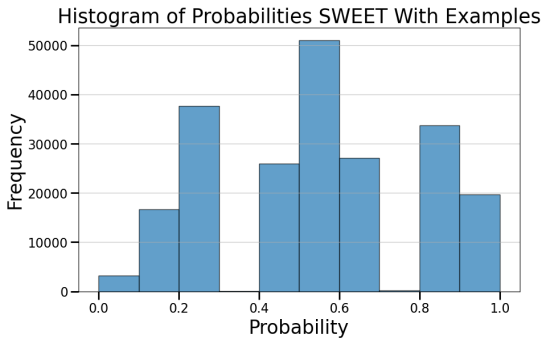
5. Discussion

5.1 Task A

The results of these experiments suggest several trends that may impact the performance of LLMs for OL. Firstly, early experiments (data not shown) indicated that



(a) *Edge probability values returned by LLMs for the SWEET dataset when not provided with examples*



(b) *Edge probability values returned by LLMs for the SWEET dataset when provided with examples*

Figure 4. *Histograms of the LLM-generated probabilities that edges existed*

Table 13. *This table shows the difference in the output with the basic and example on the test dataset.*

Dataset	F1 Score	Precision	Recall	Prompt Type
SWEET	0.1448	0.1028	0.245	Without Examples
SWEET	0.0604	0.0336	0.3046	With Examples

elaborate prompting strategies negatively impacted performance, possibly by providing conflicting cues to the LLMs. We found that simpler prompts that defined pattern-like rules performed better. Secondly, the choice between using the LLM-Centric and Classification procedures was problem-specific. The *Engineering* dataset benefited from the Classification procedure, whereas the *Scholarly* dataset generally benefited from the LLM-centric procedure. This difference may be because terms and types are more difficult to differentiate in the engineering domain, and maximizing performance requires fine-tuned, domain-specific models. Thirdly, for the Classification Procedure, the inclusion of the *neither* class generally improved F1 scores, usually by increasing precision at the cost of worse recall. Finally, we observed substantial performance disparities across specific datasets when employing a smaller model in contrast to a frontier-scale model. In the *Engineering* dataset, the frontier model vastly outperformed the smaller model, but the inverse was true in the case of the *Scholarly* dataset. While we do not understand the origin of these differences, it is possible that they are caused by differences in the training data and procedures for these models.

Across the *Scholarly* and *Engineering* datasets, we noticed a trend where our performance on the training datasets noticably outperformed performance on the testing dataset. Because the testing dataset is private, it is diffcult to draw strong conclusions based on these observations. It is possible that, even though we used simple prompts, these prompts may have "overfit" to the training data. The probability of this is increased somewhat because of the domain-specific prompts that we used. However, we also noticed some inconsistencies in the ways in which entities were represented in the training data that may also be present in the testing data. For instance, the *Engineering* dataset contained some exceptionally long entities that would be difficult to extract automatically, such as:

```
oneDistinctSymbolChangeOrSignallingEventMadeToTheTransmission-
MediumPerSecondInADigitallyModulatedSignalOrALineCode
```

and

```
luminous intensity, in a given direction, of a source that emits
monochromatic radiation of frequency 540e12 hertz and that has a
radiant intensity in that direction of 1/683 watt per steradian
```

Additionally, we observed several inconsistencies between the training documents and the provided entities in the *Scholarly* dataset. For example, terms like `cjk_compound`, `in house register`, and `non finite` appear in the provided terms, whereas their counterparts in the documents are expressed as `cjk compound`, `in-house register`, and `non-finite`. Similar discrepancies were found in the ground-truth types; for instance, `clausal arg` and `post positive arg` appear in the training dataset target, but the documents contain the forms `clausal argument` and `post positive argument`. These minor mismatches, while understandable from a human perspective, can impact model alignment and evaluation.

5.2 Task D

For Task D, we took the general approach of assuming that all possible edges existed for all nodes, and then we focused on ways to reduce the number of potential edges. While we can assume all edges exist and get an ideal recall score, we would like to discover good ways to reduce the number of edges that we assume. To this end, we investigated several methods for reducing the number of edges. Firstly, for the naive approach, we could simply provide every potential edge to an LLM and request a probability that the edge exists. This, however, gives result in about $n * n * e$ requests that need to be fulfilled by the LLM, where n is the number of nodes and e is the number of edges. While this may be feasible for a smaller dataset, like *SWEET*, which has 662 nodes and 3 edge types, it quickly becomes unmanageable with *FoodOn*, which has 2,838 nodes and 6 edge types and even more unmanageable with the *GO* dataset, containing 9,622 nodes and 6 edge types.

We used a couple of different methods for trying to reduce the number of relationships that we needed the LLM to consider: clustering, which reduces the number of requests by a factor of the number of clusters, and vector database nearest neighbors, which had the ability to bring the number of requests to $O(ne)$. While we had luck controlling the precision loss with clustering, the vector database method seemed to need a large number of neighboring nodes before it was able to attain the same level of precision. Many of the types used in the *FoodOn* and *GO* datasets are scientific names of specific things. These types may be too specialized for an off-the-shelf vector database to specify well enough. Creating a vector database with data pertaining to these datasets may help improve the results of this method. We may be able to improve clustering methods further by optimizing the number of clusters that LLMs should try to use.

We found that, by asking the LLM to output probabilities that a given edge exists, we could significantly improve the recall and F1 scores. This is not surprising, as we would expect that, by removing the least likely edges, we would do better on the recall and F1 scores. An interesting note, and something we could work on optimizing in the future, is that the distribution of the probabilities returned by the LLM seems directly related to the examples given. When more examples are given in the range over 80%, the LLM returned more predicted probabilities over 80%. Knowing this, we could determine statistics from the training set, i.e. we know that the graph is generally sparse in these sets and that there is often only one edge attached to a given node, and use this to inform the examples or prompt given. In this case, we would focus on giving lots of negative examples, as we'd expect most of our generated edges not to exist.

There are many other ways to reduce the number of potential edges to reduce the load on predicting the edge probabilities. Methods like the vector databases could be fine-tuned to work better with certain technical data. Using keywords in the dataset, like "obsolete" in *FoodOn* or *GO* can help inform what edges are likely to be seen with those nodes. We looked mostly at reducing the number of node pairs, but reducing the expected edges between certain pairs could also reduce the number of edges we need to specify.

6. Conclusions

The LLMs4OL challenge attempts to break ontology learning into a set of manageable tasks that can be accomplished by LLMs, complete with data to test methods with. This paper considers the first and last task of the LLMs4OL challenge, Task A: Term and Type Extraction and Task D: Non-taxonomic Relationship Extraction.

For Task A, our findings indicate that, for extracting terms and types: 1) prompts should be kept simple; 2) for certain datasets, a multi-step approach may improve results by extracting terms and types simultaneously and then classifying them; 3) for classification-based term and type extraction, results are improved by adding a *neither* class that helps remove incorrectly extracted terms and types; and 4) the LLM choice should be influenced by the documents' domain.

For Task D, we used methods to reduce the number of likely edges by clustering or by using vector databases. We then asked an LLM to determine the probability that each edge existed, using only clustered nodes or nodes that were nearest neighbors within the vector database. When providing LLMs with examples, the LLMs produced estimates of edge probabilities that were similar in distribution to provided examples, leading to lower performance. Additionally, the vector database methods seemed to need a large number of nearest neighbors to get good scores. Therefore, we focused on the filtering via clustering and LLM prompting without examples. We were able to run this on the SWEET dataset and the FoodOn dataset, although the SWEET dataset did much better than the FoodOn in training and testing.

Data availability statement

The data used in this study are publicly available and can be accessed at the following repository: [39]. The dataset includes training and testing data relevant to the tasks addressed in this research.

We are in the process of releasing the code. As a Department of Energy National Laboratory, Sandia National Laboratories must meet federal guidelines for asserting copyright before publicly releasing code. Interested parties should contact the authors to be notified when code is released. Once approved, code will be available at https://github.com/sandialabs.

Author contributions

Ryan Roche: (Task A) Conceptualization, Formal analysis, Investigation, Methodology, Software, Writing – original draft, Writing – review & editing

Ryn Gray: (Task D) Conceptualization, Formal analysis, Investigation, Methodology, Software, Writing – original draft, Writing – review & editing

Jaimie Murdock: Supervision, Writing – original draft, Writing – review & editing

Douglas C. Crowder: Conceptualization, Project administration, Supervision, Writing – review & editing

Competing interests

The authors declare that they have no competing interests.

Funding

This work was supported by the Laboratory Directed Research and Development program at Sandia National Laboratories, a multimission laboratory managed and operated by National Technology and Engineering Solutions of Sandia LLC, a wholly owned subsidiary of Honeywell International Inc. for the U.S. Department of Energy's National Nuclear Security Administration under contract DE-NA0003525. SAND2025-09720C

Acknowledgements

The authors acknowledge Brendan Donohoe and Ryan Dellana for their helpful feedback.

References

[1] L. Ouyang et al., *Training language models to follow instructions with human feedback*, 2022. arXiv: 2203.02155 [cs.CL]. [Online]. Available: https://arxiv.org/abs/2203.02155.

[2] T. Brown et al., "Language models are few-shot learners", *Advances in neural information processing systems*, vol. 33, pp. 1877–1901, 2020.

[3] H. Touvron et al., "Llama: Open and efficient foundation language models", *arXiv preprint arXiv:2302.13971*, 2023.

[4] G. Team et al., *Gemma: Open models based on gemini research and technology*, 2024. arXiv: 2403.08295 [cs.CL]. [Online]. Available: https://arxiv.org/abs/2403.08295.

[5] J. Wei et al., *Chain-of-thought prompting elicits reasoning in large language models*, 2023. arXiv: 2201.11903 [cs.CL]. [Online]. Available: https://arxiv.org/abs/2201.11903.

[6] K. M. Jablonka, P. Schwaller, A. Ortega-Guerrero, and B. Smit, "Leveraging large language models for predictive chemistry", *Nature Machine Intelligence*, vol. 6, no. 2, pp. 161–169, 2024.

[7] H. Inan et al., *Llama guard: Llm-based input-output safeguard for human-ai conversations*, 2023. arXiv: 2312.06674 [cs.CL]. [Online]. Available: https://arxiv.org/abs/2312.06674.

[8] P. Lewis et al., "Retrieval-augmented generation for knowledge-intensive nlp tasks", *Advances in neural information processing systems*, vol. 33, pp. 9459–9474, 2020.

[9] G. Izacard and E. Grave, "Leveraging passage retrieval with generative models for open domain question answering", *arXiv preprint arXiv:2007.01282*, 2020.

[10] M.-V. Nguyen et al., *Direct evaluation of chain-of-thought in multi-hop reasoning with knowledge graphs*, 2024. arXiv: 2402.11199 [cs.CL]. [Online]. Available: https://arxiv.org/abs/2402.11199.

[11] J. Y. Tsao et al., "Ai for technoscientific discovery: A human-inspired architecture", *Journal of Creativity*, vol. 34, no. 2, p. 100 077, 2024.

[12] J. Murdock, C. Buckner, and C. Allen, "Evaluating dynamic ontologies", in *International Joint Conference on Knowledge Discovery, Knowledge Engineering, and Knowledge Management*, Springer, 2010, pp. 258–275.

[13] Y. Rebboud, P. Lisena, L. Tailhardat, and R. Troncy, "Benchmarking llm-based ontology conceptualization: A proposal", in *ISWC 2024, 23rd International Semantic Web Conference*, 2024.
[14] 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.
[15] S. Pan, L. Luo, Y. Wang, C. Chen, J. Wang, and X. Wu, "Unifying large language models and knowledge graphs: A roadmap", *IEEE Transactions on Knowledge and Data Engineering*, vol. 36, no. 7, pp. 3580–3599, 2024. DOI: 10.1109/TKDE.2024.3352100.
[16] D. Li and F. Xu, *Synergizing knowledge graphs with large language models: A comprehensive review and future prospects*, 2024. arXiv: 2407.18470 [cs.IR]. [Online]. Available: https://arxiv.org/abs/2407.18470.
[17] L. Yang, H. Chen, Z. Li, X. Ding, and X. Wu, "Give us the facts: Enhancing large language models with knowledge graphs for fact-aware language modeling", *IEEE Transactions on Knowledge and Data Engineering*, vol. 36, no. 7, pp. 3091–3110, 2024.
[18] K. Cheng, N. K. Ahmed, R. A. Rossi, T. Willke, and Y. Sun, "Neural-symbolic methods for knowledge graph reasoning: A survey", *ACM Transactions on Knowledge Discovery from Data*, vol. 18, no. 9, pp. 1–44, 2025.
[19] L. N. DeLong, R. F. Mir, and J. D. Fleuriot, "Neurosymbolic ai for reasoning over knowledge graphs: A survey", *IEEE Transactions on Neural Networks and Learning Systems*, 2024.
[20] R. Du, H. An, K. Wang, and W. Liu, *A short review for ontology learning: Stride to large language models trend*, 2024. arXiv: 2404.14991 [cs.IR]. [Online]. Available: https://arxiv.org/abs/2404.14991.
[21] O. Perera and J. Liu, "Exploring large language models for ontology learning", 2024.
[22] S. Tsaneva, S. Vasic, and M. Sabou, "Llm-driven ontology evaluation: Verifying ontology restrictions with chatgpt", *The Semantic Web: ESWC Satellite Events*, vol. 2024, 2024.
[23] A. S. Lippolis et al., *Ontology generation using large language models*, 2025. arXiv: 2503.05388 [cs.AI]. [Online]. Available: https://arxiv.org/abs/2503.05388.
[24] T. Baldazzi et al., "Explaining enterprise knowledge graphs with large language models and ontological reasoning", in *The Provenance of Elegance in Computation-Essays Dedicated to Val Tannen (2024)*, Schloss Dagstuhl–Leibniz-Zentrum für Informatik, 2024, pp. 1–1.
[25] R. Amini, S. S. Norouzi, P. Hitzler, and R. Amini, "Towards complex ontology alignment using large language models", in *International Knowledge Graph and Semantic Web Conference*, Springer, 2024, pp. 17–31.
[26] P. Mateiu and A. Groza, "Ontology engineering with large language models", in *2023 25th International Symposium on Symbolic and Numeric Algorithms for Scientific Computing (SYNASC)*, 2023, pp. 226–229. DOI: 10.1109/SYNASC61333.2023.00038.
[27] R. M. Bakker, D. L. Di Scala, and M. H. de Boer, "Ontology learning from text: An analysis on llm performance", in *Proceedings of the 3rd NLP4KGC International Workshop on Natural Language Processing for Knowledge Graph Creation, colocated with Semantics*, 2024, pp. 17–19.
[28] N. Fathallah, S. Staab, and A. Algergawy, *Llms4life: Large language models for ontology learning in life sciences*, 2024. arXiv: 2412.02035 [cs.AI]. [Online]. Available: https://arxiv.org/abs/2412.02035.
[29] H. B. Giglou, J. D'Souza, and S. Auer, "Llms4ol: Large language models for ontology learning", *arXiv preprint arXiv:2307.16648*, 2023. DOI: 10.48550/arXiv.2307.16648. [Online]. Available: https://arxiv.org/abs/2307.16648.
[30] 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. DOI: 10.48550/arXiv.2409.10146. [Online]. Available: https://arxiv.org/abs/2409.10146.
[31] P. Kumar Goyal, S. Singh, and U. Shanker Tiwary, "Silp_nlp at llms4ol 2024 tasks a, b, and c: Ontology learning through prompts with llms", *Open Conference Proceedings*, vol. 4,

pp. 31–38, Oct. 2024. DOI: 10.52825/ocp.v4i.2485. [Online]. Available: https://www.tib-op.org/ojs/index.php/ocp/article/view/2485.

[32] M. Sanaei, F. Azizi, and H. Babaei Giglou, "Phoenixes at llms4ol 2024 tasks a, b, and c: Retrieval augmented generation for ontology learning", *Open Conference Proceedings*, vol. 4, pp. 39–47, Oct. 2024. DOI: 10.52825/ocp.v4i.2482. [Online]. Available: https://www.tib-op.org/ojs/index.php/ocp/article/view/2482.

[33] H. B. Giglou, J. D'Souza, A. C. Aioanei, N. Mihindukulasooriya, and S. Auer, *Llms4ol 2025: Large language models for ontology learning*, 2025. [Online]. Available: https://sites.google.com/view/llms4ol2025/home?authuser=0.

[34] MistralAI, *Mistral small 3.1*, Mar. 2025. [Online]. Available: https://mistral.ai/news/mistral-small-3-1.

[35] C. Greyling, *Prompt tuning, hard prompts & soft prompts*, Jul. 2023. [Online]. Available: https://cobusgreyling.medium.com/prompt-tuning-hard-prompts-soft-prompts-61b6e3e2d0b3.

[36] T. Phuttaamart, N. Kertkeidkachorn, and A. Trongratsameethong, "The ghost at llms4ol 2024 task a: Prompt-tuning-based large language models for term typing", *Open Conference Proceedings*, vol. 4, Oct. 2024. DOI: 10.52825/ocp.v4i.2486.

[37] R. Nordquist, *What is syndeton?*, 2019. [Online]. Available: https://www.thoughtco.com/what-is-syndeton-1692187.

[38] A. Rojo-Echeburúa, *Few-shot prompting: Examples, theory, use cases*, Accessed: 2025-07-07, 2024. [Online]. Available: https://www.datacamp.com/tutorials/few-shot-prompting-examples-theory-use-cases.

[39] S. Organization, *Llms4ol challenge 2025 data*, https://github.com/sciknoworg/LLMs4OL-Challenge/tree/main/2025, Accessed: 2025-07-08, 2025.