LLMs4OL 2024: The 1st Large Language Models for Ontology Learning Challenge at the 23rd ISWC
LLMs4OL 2024 Task Participant Papers
https://doi.org/10.52825/ocp.v4i.2485

Published: 02 Oct. 2024

silp_nlp at LLMs4OL 2024 Tasks A, B, and C: Ontology Learning through Prompts with LLMs

Pankaj Kumar Goyal[1], Sumit Singh[1], and
Uma Shanker Tiwary[1]

[1]Indian Institute of Information Technology, Allahabad

*Correspondence: Sumit Singh, sumitrsch@gmail.com

Abstract: Our team, *silp_nlp*, participated in the LLMs4OL Challenge at ISWC 2024, engaging in all three tasks focused on ontology generation. The tasks include predicting the type of a given term, extracting a hierarchical taxonomy between two terms, and extracting non-taxonomy relations between two terms. To accomplish these tasks, we used machine learning models such as random forest, logistic regression and generative models for the first task and generative models such as llama-3-8b-instruct, mistral 8*7b and GPT-4o-mini for the second and third tasks. Our results showed that generative models performed better for certain domains, such as subtasks A6 and B2. However, for other domains, the prompt-based technique failed to generate promising results. Our team achieved first place in six subtasks and second place in five subtasks, demonstrating our expertise in ontology generation.

Keywords: Large Language Models, LLMs, Ontology Learning, Prompt-based Learning, GPT, Llama

1 Introduction

Ontology Learning (OL) is essential in artificial intelligence as it enables the automatic extraction and organization of knowledge from text data. Traditional methods of creating ontologies often require manual input from domain experts, resulting in a time-consuming and costly process. Recent progress in natural language processing, especially through Large Language Models (LLMs), offers a compelling alternative for automating this procedure.

The LLMs4OL paradigm, as explained in the [1], aims to use large language models (LLMs) to improve tasks in ontology learning (OL), such as term typing, discovering taxonomies, and extracting non-taxonomic relations. These models, trained on a large amount of text data, can understand complex language patterns, which can be useful for building ontologies. This study expands on this idea by using similar methods in new areas to further prove that LLMs can help automate ontology learning and reduce the need for manual input. The work in this paper extends the mentioned task to cover fifteen subtasks, as listed in table 3 and details of all tasks described in [2]. We utilized machine learning models, such as random forest and logistic regression, as

well as generative models for Task A, which involves predicting the type of a given term across multiple domains. In Task B, our objective was to find taxonomies for a given pair of terms, also across multiple domains. Task C is similar to Task B, where the goal was to find non-taxonomic relations. For Tasks B and C, we employed a prompt-based technique with generative models such as mistral 8*7b [3], llama-3-8b-instruct [4] and gpt-4o [5] for taxonomy prediction. Our results showed that generative models performed better for certain domains, such as subtasks A6 and B2. However, for other domains, the prompt-based technique failed to generate promising results. The code of our work is available here[1].

2 Related Work

The OL has been a major focus of research in the fields of artificial intelligence and knowledge engineering. It aims to automate the process of acquiring and structuring knowledge from text to create ontologies [6]. Traditionally, this process has relied on manual efforts by domain experts, which can be time-consuming, costly, and error-prone. To overcome these challenges, various approaches have been proposed to automate OL, primarily using lexico-syntactic pattern mining and clustering techniques [6]–[9].
One of the earliest significant works in the field of ontology learning, as highlighted by [10], involved using lexico-syntactic patterns to improve lexical ontologies like WordNet by extracting new lexicosemantic concepts and relations from large unstructured text collections. Following approaches, such as those by Hwang (2002), used iterative methods to discover types and taxonomies from text using seed terms, while [11] focused on expanding existing ontologies by reusing domain-specific ones and integrating verb patterns from text.

In recent years, developments in Ontology Learning have involved the use of machine learning techniques. Methods such as the self-organizing tree algorithm have been utilized to create hierarchical structures within ontologies. [12] introduced a TF-IDF-based term classifier and pattern finder to automatically identify domain-specific terms and relations from text. This demonstrates the evolving nature of OL methodologies.

The field of online learning has undergone significant changes with the development of Large Language Models (LLMs). These models, which are trained on extensive and diverse text collections, have shown potential in understanding complex language patterns and have been used to explore new approaches in online learning. The LLMs4OL method, introduced by [1], examines the idea that LLMs can effectively use their language modelling abilities for online learning tasks such as identifying terms, discovering taxonomies, and extracting relationships. This method demonstrates the potential of LLMs in overcoming the limitations of traditional online learning approaches, especially when customized for specific areas.

Despite the progress made, the research suggests that basic LLMs may not be skilled enough at complex ontology construction that requires deep reasoning and domain expertise. However, ongoing improvements and adjustments to LLMs for ontology learning tasks continue to demonstrate potential, providing a scalable and efficient alternative to traditional techniques.

[1] https://drive.google.com/drive/folders/1vRynlNH6LouIvcI1ymHsm6DwYKSOUoAa?usp=sharing

3 Datasets

The organizers of the event have provided the dataset for each subtask. The details of the dataset can be described in [13]. Some subtasks do not have training data, and our goal is to develop zero-shot (ZS) solutions. However, training data is available for specific subtasks which require a few-shot (FS) approach. A list of all the subtask and Their statistics for all datasets of Task A and Task B are tabulated in Table 1 and Table 2, respectively. Tasks A and B are divided into various subtasks according to various domains. For example, the dataset for subtask A1 was taken from Wordnet. Also, we can see that for task A, the number of classes varies. For example, subtask A1 has four classes, and subtask A4 has 792 classes.

Table 1. *Table shows the size of training data, testing data and number of classes for each subtask of Task A.*

Task	Training Data	Testing Data	Number of classes
A.1(FS) - WordNet	40,559	9,470	4
A.2(FS) - GeoNames	8,078,865	702,510	680
A.3(FS) - UMLS(NCI)	96,177	24,045	125
A.3(FS) - UMLS(MEDCIN)	277,028	69,258	87
A.3(FS) - UMLS(SNOMEDCT_US)	278,374	69,594	125
A.4(FS) - GO(Biological Process)	195,775	108,300	792
A.4(FS) - GO(Cellular Component)	228,460	126,485	323
A.4(FS) - GO(Molecular Function)	196,074	107,432	401
A.5(ZS)	-	44,724	484
A.6(ZS)	-	18,078	12

Table 2. *Table shows the size of training and testing data of each subtask of Task B.*

Task	Training Data	Test Data
B.1(FS) - GeoNames	476	204
B.2(FS) - Schema.org	1,070	364
B.3(FS) - UMLS	74	45
B.4(FS) - GO	33,703	5,753
B.5(ZS)	-	762

4 Methodology

4.1 Methodology for Task A (Term Typing)

Term typing is a fundamental task in Natural Language Processing (NLP) that involves categorizing terms or words into predefined types or categories based on their semantic meaning, context, attributes, and relationships with other terms. The subtasks of task A involve two types: zero-shot and few-shot.

4.1.1 Methodology for the few-shot subtasks (A1 to A4)

For the few-shot subtasks, we trained models using machine learning algorithms such as random forest, logistic regression, and XGBoost. Each term was converted into embeddings using the tf-idf model, where the size of each vector is equal to the total number of unique terms in the training and testing datasets.

4.1.2 Methodology for the zero-shot subtasks (A5 and A6)

We have used two approaches. In the first approach, we have utilized bert [14] and sentence transformer models for the features extraction of the terms and types, and thereafter, we calculate cosine similarity between a term with all types. Most similar types are predicted as types of the term.

In the second approach, we prompted our query to the generative models. our best results for the A.6(ZS) achieved with lama-3-8b- instruct model with the following prompt:

Prompt:

system_prompt = f"""Term typing involves Categorize terms into predefined types or categories based on their attributes. Return the answer as JSON, with each term as a key and its corresponding type from the available types as the value."""

user_prompt = f"""term:{term},term definition:definition. classify the given term into one of types:[{list of categories or types}]"""

assistant_prompt = """{"area of barren land":"Environment"}"""

In above prompt we have provided term and some information about term (information about the term are extracted with the model in advance.) with list of types and asked model to find the type of term from the given list of types. Assistant prompt is showing an example with a specific output format.

4.2 Methodology for Task B (Taxonomy Discovery)

Taxonomy discovery is a task in which we need to identify the hierarchical relationship between type pairs. In this task, instances T_a and T_b are given, where Ta is the superclass (parent) of Tb, and Tb is the subclass (child) of Ta. This represents the taxonomy relationship between the two types. This task was also divided into two types of subtasks: zero-shot and few-shot subtasks.

4.2.1 Methodology for the few-shot subtasks (B1, B2 and B3)

In this task, we're given training data with term types and corresponding taxonomy-related type tuples. In the testing data, only the term types are provided, and we must identify the correct taxonomic relationships from those terms. We have used few-shot prompting in multiple ways to predict the relation.

Few-Shot Prompting through Description-Based Approaches with GPT-4o

First, find the description of each term. Afterwards, we provided a pair of terms with descriptions, along with a list of possible relations to GPT-4o and asked to select the most suitable relation between the given terms. We have also provided some examples of pairs and their relation so that the model can understand the task with the example. To maintain efficient performance.

Few-shot Prompting with GPT-4o

This method is applicable for small dataset. In this approach, a list of all the terms is provided to the GPT-4o and asked to find the pairs from the given list of terms which have hierarchical relation. We have also provided some examples of pairs and their

relation so that the model can understand the task with the example. An example of prompting for the B.2(FS)-schema.org subtask with gpt-4o model is:

Prompt:

{"role":"system","content":"""extract all the terms having parent child relationship means superclass subclass and return answer as a list of dict where the list contain all parent child relationship and dict contains keys as the parent and child and value of keys the parent and child which are possible from the given list of terms. return answer like this {"parent":Animal, "child:"elephant"}"""}, {"role": "user", "content": f"Here is the list of terms :{test data}"}

Verification-based Few-shot Prompting with mistral-22-7b

We provided a pair of terms, along with a list of possible relations to mistral-22-7b and asked to select the most suitable relation between the given terms. Thereafter, we instruct the model to verify the relation. We have also provided some examples of pairs and their relation so that the model can understand the task with the example.

4.3 Methodology for Task C1 (FS) UMLS (Non-Taxonomic Relationship Extraction)

Task C is similar to Task B, except that the terms do not have a hierarchical taxonomy. We have utilized the gpt4o for prompting. For the prediction, all combinations of pairs are provided to the model and asked whether each term of a pair is related or not. We have also provided some examples of pairs and their relation so that the model can understand the task with the example.

5 Evaluation Metric

For Task A, the precision and f1-score are reported as the metrics for the task. We have reported the same metrics. Similarly, evaluations for Task B are reported in terms of the standard F1-score based on precision and recall.

6 Results and Analysis

Our best results are tabulated in Table 3 with their respective ranks. For subtasks A1, A2, A3, and A4, a comparison of results with the random forest, logistic regression, and XGboost models is shown in Table 4. The Random forest model achieved better scores, but it required more training time compared to logistic regression and XGboost.

Similarly, for subtasks A5 and A6, which are zero-shot tasks, the results with various models are shown in Table 5. The results show that GPT-4o performed better using a prompting-based approach, whereas the sentence transformer performed more effectively with a similarity-based approach.

For subtasks B and C, results with various generative models are tabulated in Table 6. The GPT-4o model demonstrated better performance for the relation extraction tasks. However, the results of B.1(FS)-GeoNames and C.1(FS)-UMLS subtasks are still challenging since no model produces good results for these subtasks.

Table 3. *Table displays our highest F1-scores and rankings for all subtasks.*

Task	F1-score	Precision	Recall	Rank
A.1(FS) - WordNet	0.90	0.90	0.90	6
A.2(FS) - GeoNames	0.44	0.75	0.31	2
A.3(FS) - UMLS(NCI)	0.69	0.87	0.57	2
A.3(FS) - UMLS(MEDCIN)	0.93	0.95	0.92	1
A.3(FS) - UMLS(SNOMEDCT_US)	0.75	0.85	0.67	2
A.4(FS) - GO(Biological Process)	0.26	0.40	0.20	1
A.4(FS) - GO(Cellular Component)	0.27	0.42	0.20	1
A.4(FS) - GO(Molecular Function)	0.29	0.41	0.23	1
A.5(ZS)	0.30	0.30	0.30	2
A.6(ZS)	0.72	0.72	0.72	2
B.1(FS) - GeoNames	0.08	0.04	0.59	3
B.2(FS) - Schema.org	0.61	0.45	0.94	1
B.3(FS) - UMLS	0.35	0.41	0.31	1
B.5(ZS)	0.21	0.14	0.42	1
C.1(FS) UMLS	0.07	0.04	0.18	1

Table 4. *Comparison of F1-score across different machine learning models for few-shot subtasks of task A.*

Task Name	Random Forest	Logistic Regression	XGboost
A.1(FS) - WordNet	0.9037	0.68	0.69
A.2(FS) - GeoNames	0.4433	0.31	0.40
A.3(FS) - UMLS(NCI)	0.6973	0.4706	-
A.3(FS) - UMLS(MEDCIN)	0.9381	-	-
A.3(FS) - UMLS(SNOMEDCT_US)	0.7552	0.7334	0.7552
A.4(FS) - GO(Cellular Component)	-	0.2725	-
A.4(FS) - GO(Biological Process)	0.24	0.2349	0.269075
A.4(FS) - GO(Molecular Function)	0.20	0.267	0.297

7 Conclusion

During our investigation, we explored different machine learning and generative models for ontology generation as part of the LLMs4OL Challenge @ ISWC 2024. Our approach involved using traditional machine learning models such as Random Forest, Logistic Regression, and XGBoost, as well as advanced generative models like llama-3-8b-instruct, mistral 8*7b, and GPT-4o.

Our results showed that different approaches had varying effectiveness across tasks. For subtasks A1 through A4, Random Forest models yielded superior results, although they required longer training times compared to Logistic Regression and XGBoost. For zero-shot tasks A5 and A6, GPT-4O proved to be the most effective model, highlighting the potential of advanced generative models in scenarios where labelled data is limited. Similarly, for subtasks B and C, which focused on relation extraction, GPT-4O also outperformed other models, demonstrating its suitability for complex NLP tasks.

Table 5. *Comparison of F1-score across different models for zero-shot subtasks of task A.*

Task Name	bert-base-uncased	sentence-transformers/all-MiniLM-L6-v2	mistral 8*7b	GPT-4o-mini	llama-3-8b-instruct
A.5(ZS)	0.146	0.2001	0.2906	0.3008	-
A.6(ZS)	0.30	0.39	-	-	0.7278

Table 6. *Comparison of the F1-scores across different models for tasks B and C.*

Task Name	llama3-8b-fine-tuning-predibase	llama3-8b	gpt-4o	mistral 22*7b
B.1(FS) - GeoNames	0.083	0.041	-	-
B.2(FS) - Schema.org	-	-	0.61	-
B.3(FS) - UMLS	-	-	0.3544	0.1834
B.5(ZS)	-	-	0.2109	-
C.1(FS)-UMLS	-	0.047	0.0616	-

Author contributions

Pankaj Kumar Goyal: Data curation, Methodology, Validation, Implementation.
Sumit Singh: Conceptualisation, Writing – Original Draft, Writing – Review & Editing, Investigation.
Uma Shanker Tiwary: Supervision.

Competing interests

The authors declare that they have no competing interests.

References

[1] H. Babaei Giglou, J. D'Souza, and S. Auer, "Llms4ol: Large language models for ontology learning," in *The Semantic Web – ISWC 2023*, T. R. Payne, V. Presutti, G. Qi, *et al.*, Eds., Cham: Springer Nature Switzerland, 2023, pp. 408–427, ISBN: 978-3-031-47240-4.

[2] H. Babaei Giglou, J. D'Souza, and S. Auer, "Llms4ol 2024 overview: The 1st large language models for ontology learning challenge," *Open Conference Proceedings*, vol. 4, Oct. 2024.

[3] A. Q. Jiang, A. Sablayrolles, A. Mensch, *et al.*, *Mistral 7b*, 2023. arXiv: 2310.06825 [cs.CL]. [Online]. Available: https://arxiv.org/abs/2310.06825.

[4] A. Dubey, A. Jauhri, A. Pandey, *et al.*, *The llama 3 herd of models*, 2024. arXiv: 2407.21783 [cs.AI]. [Online]. Available: https://arxiv.org/abs/2407.21783.

[5] openai. "Gpt-4o." (2024), [Online]. Available: https://openai.com/index/hello-gpt-4o/.

[6] A. Konys, "Knowledge repository of ontology learning tools from text," *Procedia Computer Science*, vol. 159, pp. 1614–1628, 2019, Knowledge-Based and Intelligent Information Engineering Systems: Proceedings of the 23rd International Conference KES2019, ISSN: 1877-0509. DOI: https://doi.org/10.1016/j.procs.2019.09.332. [Online]. Available: https://www.sciencedirect.com/science/article/pii/S1877050919315339.

[7] C. Fellbaum and G. Miller, "Automated discovery of wordnet relations," in *WordNet: An Electronic Lexical Database*. 1998, pp. 131–151.

[8] C. H. Hwang, "Incompletely and imprecisely speaking: Using dynamic ontologies for representing and retrieving information," in *Knowledge Representation Meets Databases*, 1999. [Online]. Available: https://api.semanticscholar.org/CorpusID:11502906.

[9] L. Khan and F. Luo, "Ontology construction for information selection," in *14th IEEE International Conference on Tools with Artificial Intelligence, 2002. (ICTAI 2002). Proceedings.*, 2002, pp. 122–127. DOI: 10.1109/TAI.2002.1180796.

[10] Z. Akkalyoncu Yilmaz, S. Wang, W. Yang, H. Zhang, and J. Lin, "Applying BERT to document retrieval with birch," in *Proceedings of the 2019 Conference on Empirical Methods in Natural Language Processing and the 9th International Joint Conference on Natural Language Processing (EMNLP-IJCNLP): System Demonstrations*, S. Padó and R. Huang, Eds., Hong Kong, China: Association for Computational Linguistics, Nov. 2019, pp. 19–24. DOI: 10.18653/v1/D19-3004. [Online]. Available: https://aclanthology.org/D19-3004.

[11] *OL'00: Proceedings of the First International Conference on Ontology Learning - Volume 31*, Berlin, Germany: CEUR-WS.org, 2000.

[12] F. Dalvi, A. R. Khan, F. Alam, N. Durrani, J. Xu, and H. Sajjad, "Discovering latent concepts learned in bert," *ArXiv*, vol. abs/2205.07237, 2022. [Online]. Available: https://api.semanticscholar.org/CorpusID:248810913.

[13] H. Babaei Giglou, J. D'Souza, S. Sadruddin, and S. Auer, "Llms4ol 2024 datasets: Toward ontology learning with large language models," *Open Conference Proceedings*, vol. 4, Oct. 2024.

[14] J. Devlin, M.-W. Chang, K. Lee, and K. Toutanova, "BERT: Pre-training of deep bidirectional transformers for language understanding," in *Proceedings of the 2019 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, Volume 1 (Long and Short Papers)*, J. Burstein, C. Doran, and T. Solorio, Eds., Minneapolis, Minnesota: Association for Computational Linguistics, Jun. 2019, pp. 4171–4186. DOI: 10.18653/v1/N19-1423. [Online]. Available: https://aclanthology.org/N19-1423.