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.2491

Published: 02 Oct. 2024

RWTH-DBIS at LLMs4OL 2024 Tasks A and B

Knowledge-Enhanced Domain-Specific Continual Learning and Prompt-Tuning of Large Language Models for Ontology Learning

Yixin Peng[1], Yongli Mou[1], Bozhen Zhu[1], Sulayman Sowe [12], and Stefan Decker [12]

[1]Chair of Computer Science 5, RWTH Aachen University, Aachen, Germany

[2]Fraunhofer Institute for Applied Information Technology (FIT), Sankt Augustin, Germany

*Correspondence: Yongli Mou, mou@dbis.rwth-aachen.de

Abstract: The increasing capabilities of Large Language Models (LLMs) have opened new opportunities for enhancing Ontology Learning (OL), a process crucial for structuring domain knowledge in a machine-readable format. This paper reports on the participation of the RWTH-DBIS team in the LLMs4OL Challenge at ISWC 2024, addressing two primary tasks: term typing and taxonomy discovery. We used LLaMA-3-8B and GPT-3.5-Turbo models to find the performance gaps between open-source and commercial LLMs. For open-source LLMs, our methods included domain-specific continual training, fine-tuning, and knowledge-enhanced prompt-tuning. These approaches were evaluated on the benchmark datasets from the challenge, i.e., GeoNames, UMLS, Schema.org, and the Gene Ontology (GO), among others. The results indicate that domain-specific continual training followed by task-specific fine-tuning enhances the performance of open-source LLMs in these tasks. However, performance gaps remain when compared to commercial LLMs. Additionally, the developed prompting strategies demonstrate substantial utility. This research highlights the potential of LLMs to automate and improve the OL process, offering insights into effective methodologies for future developments in this field.

Keywords: Ontology Learning, Large Language Models, Domain-specific Continual Learning, Knowledge-enhanced Prompt-tuning, Hierarchical Text Classification

1 Introduction

In the context of the Semantic Web community, an ontology is a formal representation of a set of concepts and the relationships between those concepts of shared conceptualizations of a domain of interest, shared by a group of people within a certain domain [1], [2]. It is often considered as a source of semantics and interoperability and used to model domain knowledge in a structured, machine-readable format. Ontology Learning (OL) refers to the process of automatic or semi-automatic creation of ontologies from text, including the extraction of terms and concepts, the extraction relationships, axiom and evaluation [3].

The recent success of OpenAI's Generative Pre-trained Transformer (GPT) has demonstrated the enormous capabilities of Large Language Models (LLMs) in natural language understanding and generation. LLMs have shown proficiency in various tasks, including machine translation, summarization, question-answering, and more recently, and now open new opportunities for enhancing OL processes, a domain-specific task. *LLMs4OL* [4] defines six key activities in OL including corpus preparation, terminology extraction, including term typing, taxonomy construction, relation extraction, and axiom discovery. The LLMs4OL Challenge ISWC 2024 [5] introduces three tasks of conceptualization (term typing, taxonomy construction, and relation extraction) and aims to explore and harness the potential of LLMs in OL within the context of the Semantic Web. It seeks to foster innovation and collaboration in the development of scalable and precise methods. This paper presents a technical report on our participation in the challenge.

1.1 Tasks Performed

In this study, we addressed two primary tasks defined by the LLMs4OL Challenge:

- Task A - Term Typing: Discover the generalized type for a lexical term.
- Task B - Taxonomy Discovery: Discover the taxonomic hierarchy between type pairs.

1.2 Main Objectives of Experiments

The primary objectives of our experiments were to assess the effectiveness of both open-source and commercial LLMs in tasks such as term typing and taxonomy discovery. Our research is centered around the following research questions, which are guiding the design of our experiments and the subsequent analysis of the results:

1. **Comparing the Performance of Commercial and Open-Source Models**: How do existing commercial models stack up against open-source models in terms of performance on the tasks defined in this competition? This question is aimed at evaluating the strengths and limitations of different models when applied to domain-specific challenges.
2. **Enhancing Open-Source Models through Training Methods**: Can the performance of open-source models be improved through the application of various training methods? This question explores whether targeted training strategies can narrow the performance gap between commercial and open-source models in these tasks.

2 Background and Related Work

For both tasks in the challenge, our research focuses on the following topics: Unsupervised Continual Learning and Knowledge-enhanced Prompt-tuning.

Unsupervised Continual Learning

Unsupervised continual learning (UCL) focuses on the ongoing training of models using unlabeled data, a technique that has shown considerable success across various fields. In the continual pre-training scenario, models are first pre-trained on a large corpus of general text and then further pre-trained on domain-specific data to better adapt to new domains. This method has led to significant performance improvements for tasks within new domains. For instance, Gururangan et al. [6] explored the effectiveness of domain-adaptive pre-training for NLP models. Their study demonstrated that continual pre-training on domain-specific data could significantly enhance the performance of downstream tasks, even when the amount of data is limited. Additionally, researchers such as Ke et al. [7], Scialom et al. [8], and Han et al. [9] formalized the continual pre-training scenario. They proposed pre-training strategies that effectively retain and transfer knowledge, thereby improving the generalization capabilities of the models. Their work also yielded promising results in the field of natural language processing (NLP).

The study of continual learning in large language models (LLMs) is still evolving. Wu et al. [10], wang et al. [11], and shi et al. [12] provided a comprehensive survey on continual learning for LLMs, emphasizing the necessity for regular updates to keep the models current with evolving knowledge and skills. Those studies show that continual pre-training and unsupervised continual learning offer promising avenues for enhancing the adaptability and performance of models in language domains. Therefore, we will develop our own continuous pre-training scheme based on the contextual information collected from the domain of the provided training data.

Knowledge-enhanced Prompt-tuning

The integration of external knowledge into prompt-tuning methods has shown promising results in enhancing the performance of few-shot learning models. Traditional prompt-tuning approaches often struggle with tasks requiring domain-specific knowledge due to their reliance on general-purpose language models pre-trained on vast but generic datasets [13]. To address these limitations, several studies have explored the incorporation of structured and unstructured knowledge into prompt-tuning frameworks. Lu et al. [14] proposed the Medical Knowledge-enhanced Prompt Learning (MedKPL) model to improve diagnosis classification from clinical text. MedKPL leverages both structured knowledge from medical knowledge graphs and unstructured knowledge from online resources, integrating them into the prompt templates. The model demonstrated superior performance in standard and low-resource settings, showcasing its robustness and transferability across different medical departments. Similarly, liu et al. [15] introduced the Structured Knowledge Prompt Tuning (SKPT) method, which enhances prompt learning by embedding structured knowledge directly into the prompt sequences. This approach utilizes open information extraction to generate knowledge triples, which are then incorporated into the prompt templates. SKPT has shown effectiveness in few-shot text classification tasks, outperforming traditional methods on benchmark datasets.

These advancements highlight the potential of knowledge-enhanced prompt-tuning to address the challenges of domain-specific tasks in NLP. By integrating external knowledge into prompt templates, these methods not only improve model performance but also enable better generalization to unseen data and low-resource scenarios.

3 Methods

In this section, we detail the methodologies employed in our approach to the LLMs4OL Challenge at ISWC 2024. Our approach depicted in Figure 1 is structured into three main stages data augmentation, model training, and inference. Each stage involves specific techniques and processes designed to optimize the performance of LLMs in term typing and taxonomy discovery.

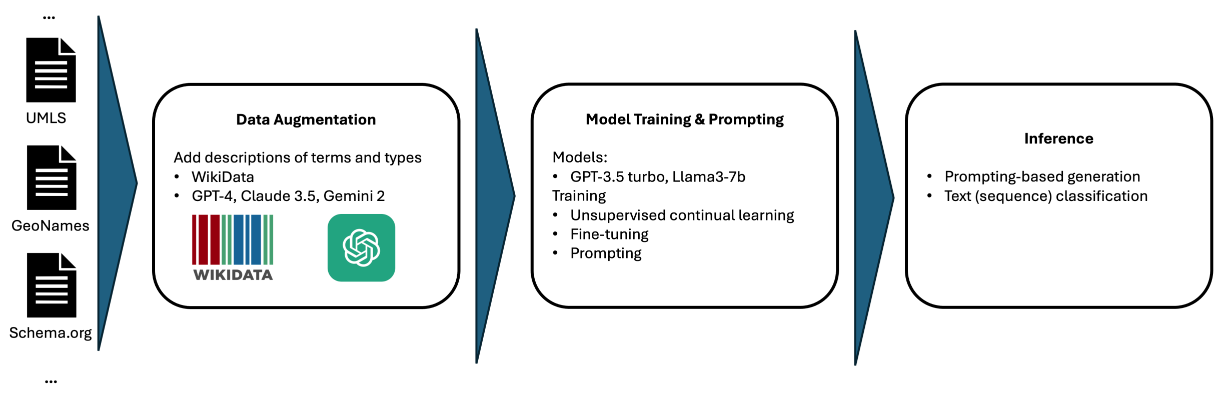
...
UMLS
GeoNames
Schema.org
...
Data Augmentation
Add descriptions of terms and types
• WikiData
• GPT-4, Claude 3.5, Gemini 2
WIKIDATA
Model Training & Prompting
Models:
• GPT-3.5 turbo, Llama3-7b
Training
• Unsupervised continual learning
• Fine-tuning
• Prompting
Inference
• Prompting-based generation
• Text (sequence) classification

Figure 1. *Overview of Research Methodology*

3.1 Data Augmentation for Terms and Types

We participated in both Task A and Task B of the competition, each of which provided a set of ontologies [16]. Task A included the ontologies **WordNet** [17], **GeoNames** [18], **UMLS** [19], and **GO** [20]. The training data for Task A comprised three components: an optional context sentence (available only for WordNet), a lexical term, and a conceptual term type (provided as a list if the term could be assigned multiple types). In contrast, the test dataset for Task A only provided the optional context sentence (again, only for WordNet) and the lexical term.

Task B included the ontologies **GeoNames** [18], **UMLS** [19], **GO** [20], and **Schema.org** [21]. The training data for Task B was structured as pairs of types in the format $\{T_a, T_b\}$, where T_a represents the parent (superclass) and T_b represents the child (subclass). The test dataset for Task B provided only a series of types, requiring us to identify pairs with an ”is-a” relationship among them.

During model training, we collected contextual information for terms and types using three methods described in this section, based on the data provided in the training datasets. Notably, for **GeoNames** [18], we gathered contextual information for all types and terms that appeared in both the Task A and Task B training datasets. For **Schema.org** [21], we collected information for all types present in both the training and test datasets of Task B. For **UMLS** [19] and **GO** [20], we collected contextual information only for the types found in the Task B training dataset. No contextual information was collected for **WordNet** [17].

3.1.1 Data Collection from Wikipedia

We search for term or type descriptions through publicly available sources, such as Wikipedia. We utilize the Wikipedia API [22] to retrieve relevant term definitions by sending HTTP requests and parsing the returned JSON data. After obtaining the raw content, we perform data cleaning to remove irrelevant or redundant information, ensuring the data is more standardized and structured. The specific cleaning steps include:

- **Filtering Parser Instructions**: Use regular expressions to match and remove magic words such as ”__NOTOC__” that control page content display and behavior but are not useful for our training purposes.
- **Filtering File/Image Links**: Remove file or image links starting with File, Image, or Media.
- **Filtering References and Tables**: Remove content within ”ref” and ”table” tags.

- **Handling Category Links**: Retain category links but remove the category prefixes.
- **Merging and Cleaning Content**: Combine various parts of the content and remove extra line breaks and whitespace characters.

This data collection and cleaning process ensures that the data obtained from Wikipedia is of high quality and consistency. The cleaned content is used for subsequent data analysis and model training. It is also suitable for collecting the definitions of various types.

3.1.2 Data Collection using Commercial Language Models

We employ commercial large language models with network search capabilities to collect a large amount of term-related information required for training, such as OpenAI’s GPT-4o [23], Anthropic’s Claude-3 [24], and Microsoft’s Copilot [25]. We use zero-shot prompting to access these models through their API interfaces, and after cleaning the returned data, it is used as training data to enrich the contextual information of terms and types. Figure 2 demonstrates the example prompt we used to collect context information for the GeoNames dataset.

Inference Prompt

Here is a geographical term: ”{name}”. Translate the given geographical term into English. Provide geographical information in plain text without any markdown formatting or reference links. Ensure that all provided information is useful for discovering implicit relationships with other geographical terms. Do not include the relationships in the result.

Figure 2. *Template used to prompt commercial LLMs to generate relevant information about terms in dataset GeoNames.*

As an example, we present a data entry from the GeoNames training dataset for a given term: ”Cascade Bluff”. Using the prompt provided above, we obtained the following response from GPT-3.5-Turbo: ”Cascade Bluff is a geographical location that can be translated into English. It is a place characterized by steep slopes or cliffs formed by the cascading of water. The term ”Cascade” refers to the manner in which water descends rapidly and forcefully, creating a series of small waterfalls or a large waterfall. ”Bluff” indicates a high, steep, or vertical cliff or bank. Geographically, Cascade Bluff is likely to be situated near a body of water, such as a river, where the cascading effect is prominent. It could be part of a hilly or mountainous region with pronounced cliffs or bluffs.”

3.1.3 Direct Access to Ontology Data Sources

We accessed the ontology data sources associated with the datasets directly through their provided APIs or by downloading their datasets. This allowed us to retrieve and store contextual information pertinent to the training data. The contextual information refers to the explanatory text descriptions of the terms and types included in the training set, as provided within the ontology datasets. For instance, in the case of the type ”lake”, we extracted the following descriptive explanation from the GeoNames [18]: ”a large inland body of standing water”. The datasets utilized include **GeoNames** [18], **UMLS** [19], **GO** [20], and **Schema.org** [21].

3.2 Model Continual Learning

3.2.1 Domain-specific Continual Training

This approach involves domain-specific continual training of the selected open-source model. We enriched the training data using the context information for terms and types obtained through the method described in Section 3.1. For the GeoNames dataset, we collected context information for all terms and types present in the training datasets of Task A and Task B. For other datasets, we collected only the context information for types.

Figure 3 we present the training prompts constructed using type information alone and using both type and term information. These texts were used to train the model in a CausalLM manner. An analysis of the types contained in the GeoNames dataset reveals that there are 9 level-1 types and 671 level-2 types. In the prompt template provided below, "L2" represents the type of the term from the training dataset, while "L1" denotes the super-type of that type.

Training Prompts

```
# Prompt using type information alone
```

Type: "{type_name}", {type_info}

```
# Prompt using both type and term information
```

The term "{term}" from the GeoNames dataset. {term_info}. It falls under the top-level classification of "{L1}". Given this description, it can be logically inferred that "{term}" should belong to the specific sub-category "{L2}" within this top-level classification."{L2}" is described as: {L2_info}. Based on this inference, the type of this term is determined to be "{L2}".

Figure 3. *Template of training prompts for Method A*

3.2.2 Fine-tuning

This method describes the fine-tuning of an open-source LLM for a specific task, aimed at addressing the downstream task of Sequence Classification. We employed different training strategies for Task A and Task B.

Task A

We started with the training data provided by the competition, which included terms and their corresponding types. The objective was to transform this data into a format suitable for model training. Specifically, we assigned a unique numerical label to each type and created a new dataset comprising these labels and their corresponding terms. We then fine-tuned the model using this processed data.

Task B

We begin with the training data provided by the challenge, which included a hierarchical parent-child relationship. First, we needed to transform these hierarchical relationships into a data format suitable for model training. Specifically, we generated textual descriptions and context information for each relationship and assigned a label of 1 to these positive samples. To enable the model to distinguish between correct and incorrect hierarchical relationships, we constructed negative samples by reversing the parent and child entities and assigning a label of 0 to these negative samples. Finally, we combined the positive and negative samples to create the final training dataset. Below are data samples from the dataset for GeoNames [18]. We then fine-tuned the model using this processed data.

Task B Data Processing

```
# Unprocessed Training Data
```

"parent": "mountain, hill, rock",
"child": "karst area"

```
# Processed Prompts
```

"{parent} is the superclass / parent / supertype / ancestor class of {child}, They are two geographical terms. '{parent}': {parent_info} '{child}': {child_info}"

"{child} is the subclass / child / subtype / descendant class of {parent}, They are two geographical terms. '{parent}': {parent_info} '{child}': {child_info}"

Figure 4. *Data Processing Example for Method B - Task B*

3.2.3 Domain-specific Continual Training Followed by Task-specific Fine-tuning

First, we performed unsupervised continual training on the specific dataset using the method described in Section 3.2.1. Next, we further fine-tuned the model for the specific task using the approach outlined in Section 3.2.2. The methods and training texts used in the phased training process were consistent with those previously described.

3.3 Prompting

We experimented with various prompts and ultimately selected the ones that demonstrated the best performance for the challenge. Below, we provide the templates for these prompts, each specifically designed for different tasks and models:

1. **Prompt template for inference using GPT-3.5, applicable to all datasets in Task A:** *"You are provided with a term: '{term}' from the {dataset name}, {short description about dataset}. Based on the information you have and can find on the internet, provide only the most likely type for this term. This type is strictly defined within the following list: {type_list}. You must only give me the type. Nothing else."* Here, {type_list} represents the set of all types that appear in the training dataset for Task A's {dataset name}.
2. **Prompt template for inference using GPT-3.5, specifically for the GeoNames dataset in Task A:** The GeoNames dataset contains 9 level-1 types and 671 level-2 types. First, GPT determines which of the 9 level-1 types the term belongs to. Then, based on the first result, GPT infers which level-2 type under the identified level-1 type the term belongs to. The prompts used for these inferences are similar to those described previously, with the key difference being the {type_list}

provided in the prompt, which varies according to the hierarchical level being inferred.

3. **Prompt template for inference using the model trained with Method A (Section 3.2.1), specifically for the GeoNames dataset in Task A:** *"The term '{term}' from the GeoNames dataset."* The generated text needs to extract the type information following the keyword "sub-category" using a regular expression, and this type information is returned as the result.
4. **Prompt template for inference using the model trained with Method B (Section 3.2.2), applicable to the UMLS and GO datasets in Task A:** Use the term directly from the test data.
5. **Prompt template for inference using the model trained with Method B (Section 3.2.2), specifically for the GeoNames dataset in Task B:** *"{parent} is the superclass of {child}."* Here, {parent} and {child} are paired combinations from the type list provided in the test data.
6. **Prompt template for inference using the model trained with Method C (Section 3.2.2), applicable to the GeoNames and Schema datasets in Task B:** Use the prompt provided in the fifth point above.

4 Evaluation

In this section, we present the evaluation of our approach to term typing and taxonomy discovery as part of the LLMs4OL Challenge at ISWC 2024. The evaluation focuses on assessing the performance of our models in accurately classifying terms and constructing taxonomic hierarchies.

4.1 Experimental Setup

The evaluation was conducted on datasets provided for the LLMs4OL challenge, consisting of diverse sets of terms and their corresponding types and hierarchical relationships, mentioned in the previous section.

We mainly use the Hugging Face *transformers* library for the implementation, which provides robust tools for training and deploying state-of-the-art large language models, and the pre-trained model from Hugging Face model repository *meta-llama/Meta-Llama-3-8B*. The model training and inference were conducted on a high-performance server equipped with four NVIDIA H100 80GB GPUs. The duration of training depends on the size of the training data. For instance, using the GeoNames dataset from Task A, which contains 8,078,865 terms and approximately 6 GB of extracted contextual information, training on 4 GPUs for 5 epochs takes around 200 hours to complete. Training hyper-parameters are the following:

- **Learning Rate**: 2×10^{-5}
- **Batch Size**: 16 samples per device
- **Number of Epochs**: 5 (for each mentioned training method)
- **Weight Decay**: 0.01
- **Mixed Precision Training**: Enabled (`fp16`)
- **Gradient Accumulation**: Accumulate gradients over 4 steps
- **Optimizer**: AdamW

The learning rate is a critical hyper-parameter that controls the step size during the update of model weights. In each back-propagation step, the model adjusts its weights according to the gradient of the loss function, with the learning rate determining the magnitude of these adjustments. Batch size refers to the number of samples input into the model during each training iteration. The number of epochs represents the number of complete passes through the training dataset. One epoch consists of a single forward and backward pass through the entire training dataset.

Weight decay is a regularization technique designed to prevent over-fitting by constraining the magnitude of model weights. It achieves this by adding the L2 norm of the weights to the loss function, thereby penalizing excessively large weights and discouraging overly complex models. The strength of this penalty is controlled by the weight decay parameter.

Mixed precision training [26] involves using 16-bit floating-point numbers (fp16) instead of 32-bit floating-point numbers (fp32) to represent model parameters and gradients during training. This approach can significantly reduce memory usage and increase computational speed with minimal impact on the final model performance.

Gradient accumulation is a technique employed to achieve larger effective batch sizes when memory resources are limited. Specifically, instead of updating the model weights after each forward pass, the gradients are accumulated over multiple iterations (in this case, four) and then used to update the weights. This method allows for the benefits of larger batch sizes, such as more stable gradient estimates, without the need for additional memory .

The AdamW optimizer [27] is a variant of the Adam optimizer [28] that improves upon the original by decoupling weight decay from the gradient update process, thereby mitigating the bias introduced by L2 regularization. AdamW combines the advantages of momentum methods and adaptive learning rate adjustments, making it particularly well-suited for training large-scale models.

4.2 Results

Table 1 and Table 2 present the evaluation results for our participation in the sub-tasks of Task A and Task B, respectively. The numbers in parentheses indicate our ranking for each metric among all participating teams in the corresponding sub-task. The results in Table 1 were obtained by evaluating GPT using the prompts provided in Section 3.3 for inference. Similarly, the results in Table 2 were derived by applying the prompts from Section 3.3 for inference on the Llama-3-8B model, which was trained using the approach described in Section 3.2.3 for Task B.

Table 1. *Results on Task A*

Subtasks	F1 Score	Precision	Recall
A.1-WordNet	0.9446 (4)	0.9446 (4)	0.9446 (4)
A.2-GeoNames	0.4355 (3)	0.4355 (3)	0.4355 (2)
A.3-UMLS (NCI)	0.1691 (4)	0.1821 (4)	0.1579 (4)
A.3-UMLS (MEDCIN)	0.4566 (4)	0.4607 (3)	0.4526 (4)
A.3-UMLS (SNOMEDCT_US)	0.4747 (4)	0.4888 (3)	0.4613 (4)
A.4-GO (Biological Process)	0.0881 (3)	0.0693 (4)	0.1207 (2)
A.4-GO (Cellular Component)	0.2178 (2)	0.1846 (2)	0.2656 (1)
A.4-GO (Molecular Function)	0.1418 (2)	0.1670 (2)	0.1231 (4)
A.5-DBpedia	0.4270 (1)	0.4270 (1)	0.4270 (1)
A.6-FoodOn	0.8068 (1)	0.8068 (1)	0.8068 (1)

Table 2. *Results on Task B Dataset*

Subtasks	F1 Score	Precision	Recall
B.1-GeoNames	0.3409 (2)	0.2400 (2)	0.5882 (3)
B.2-Schema.org	0.5733 (2)	0.5475 (1)	0.6016 (2)

Next, we will provide a detailed comparison of the performance of GPT-3.5-Turbo-0125 (referred to as G) and the Llama-3-8B model, which was trained using various methods, across the UMLS, GO, Schema.org, and GeoNames datasets. The comparative results are presented as follows:

4.2.1 Comparison of Training Methods for Task A

GeoNames Dataset

The Table 3 presents the performance comparison between two models, GPT-3.5-Turbo and Trained Llama-3-8B, on the GeoNames dataset in terms of F1 Score, Precision, and Recall. The Llama-3-8B model was trained using the method described in Section 3.2.1, which enriched the training data with context information for terms and types. Although the results show that GPT-3.5-Turbo outperforms trained Llama-3-8B across all metrics, considering the parameter sizes of the two models, it is evident that our training method is highly effective.

Table 3. *Results on Task A Dataset GeoNames*

Methods	F1	Precision	Recall
GPT-3.5-Turbo	0.4355	0.4355	0.4355
Llama3-8B	0.40396	0.40396	0.40396

UMLS and GO Datasets

Table 4 and Table 5 show that GPT-3.5-Turbo outperforms the Llama-3-8B model, fine-tuned using the method described in Section 3.2.2 for Task A, across all datasets in terms of F1 score, precision, and recall. The Llama-3-8B model, trained using only term textual information without providing context, performed poorly on downstream tasks. Additionally, the evaluation results for the UMLS dataset are generally better than those for the GO dataset. This discrepancy may be due to the fact that terms in the GO dataset can be assigned to a larger number of types, whereas our prompt method identifies only the most likely type.

Table 4. *Results on Task A for Dataset UMLS. Method G for GPT-3.5-Turbo and F for Fine-tuned Llama3-8B using Method in Section 3.2.2 for Task A*

Methods	NCI			MEDCIN			SNOMEDCT_US		
	F1	P	R	F1	P	R	F1	P	R
G	0.1691	0.1821	0.1579	0.4566	0.4607	0.4526	0.4747	0.4888	0.4613
F	0.0017	0.0018	0.0016	0.0001	0.0001	0.0001	0.0048	0.0050	0.0047

Table 5. *Results on Task A for Dataset GO. Method G for GPT-3.5-Turbo and F for Fine-tuned Llama3-8B using Method in Section 3.2.2 for Task A*

Methods	BP			CC			MF		
	F1	P	R	F1	P	R	F1	P	R
G	0.0881	0.0693	0.1207	0.2178	0.1846	0.2656	0.1418	0.167	0.1231
F	0.0022	0.0027	0.0018	0.0017	0.0021	0.0015	0.0014	0.0016	0.0011

4.2.2 Comparison of Training Methods for Task B

GeoNames and Schema.org Datasets

The Table 6 presents the performance of the same model (Llama-3-8B) trained using two different methods. Method F refers to the fine-tuning approach described in Section 3.2.2 for Task B, which is training a model using textual descriptions and context information for each relationship extracted from the training dataset, while Method P-F refers to the training approach described in Section 3.2.3 for Task B, which is domain-specific continual training followed by task-specific fine-tuning. By comparison, it is evident that the model's performance on this task significantly improves after training with Method P-F. The submitted results showed in Table 7 for the Schema.org dataset were also obtained after training with Method P-F. This method appears to enhance the model's capacity to generalize across different datasets, particularly in tasks requiring a nuanced understanding of context and relationships, as evidenced by the improved F1 scores, precision, and recall on the GeoNames and Schema.org datasets. These findings underscore the importance of incorporating domain-specific knowledge and tailored training strategies to improve the performance of large language models in specialized tasks.

Table 6. *Results for Methods F and P-F on Task B Dataset GeoNames*

Methods	F1 Score	Precision	Recall
F	0.183796	0.12199	0.372549
P-F	0.3409	0.24	0.5882

Table 7. *Results for Methods P-F on Task B Dataset Schema.org*

Methods	F1 Score	Precision	Recall
P-F	0.5733	0.5475	0.6016

4.3 Discussion

The results demonstrate that GPT-3.5-Turbo (G) outperforms the Llama-3-8B model fine-tuned for Task A across the UMLS, GO, and GeoNames datasets. This suggests that GPT-3.5-Turbo is more effective in handling the diverse and complex semantic structures present in these datasets. One possible reason for this superior performance is GPT-3.5's inherent ability to leverage a broader contextual understanding, which is crucial for accurately disambiguating terms and relationships in datasets such as UMLS and GO.

In Task B, our initial submission yielded suboptimal evaluation scores. To improve performance, we re-ran the inference process and subsequently ranked the logits scores of the outputs in descending order. Specifically, we selected the top 500 results for the GeoNames dataset and the top 400 results for the Schema.org dataset. Upon resubmission, we observed a marked improvement in the evaluation scores. This enhancement is likely due to our revised approach to handling the output data. During the inference process, we paired the types provided in the test dataset in all possible combinations, similar to constructing a matrix. However, because the ground truth labels are sparsely distributed within this matrix, the model we trained produced a relatively high number of false positives (FP) in its predictions. By refining our submission to include only the top-ranked results, we effectively reduced the number of FPs, which led to an overall improvement in the evaluation metrics. The accompanying Table 8 and Table 9) present a comparison of the evaluation results before and after implementing this top-ranking selection approach.

Table 8. *Results for All and Top 500 Predictions on GeoNames*

Methods	F1 Score	Precision	Recall
All	0.2125	0.1226	0.7941
Top	0.3409	0.24	0.5882

Table 9. *Results for All and Top 400 Predictions On Schema.org*

Methods	F1 Score	Precision	Recall
All	0.4653	0.3299	0.7890
Top	0.5733	0.5475	0.6016

Another notable observation from the results is the discrepancy in performance between the UMLS and GO datasets. GPT-3.5-Turbo achieves better evaluation metrics on the UMLS dataset compared to the GO dataset. This difference could be attributed to the nature of the datasets themselves. UMLS terms tend to have fewer associated types, making it easier for models to accurately classify them. In contrast, GO terms can be assigned to multiple types, leading to a more challenging classification task. Our prompt-based method, which identifies the most likely type, may not be fully equipped to handle the multiplicity of types in the GO dataset, resulting in lower performance metrics. This suggests that future work should explore prompt engineering techniques or model architectures that can more effectively address multi-label classification tasks.

5 Future Work

Building on the findings of this study, we offer the following recommendations for future research and development. These suggestions aim to enhance the performance of open-source LLMs in the two tasks addressed in this work:

Comprehensive Context Gathering for Task A

For future studies of Task A, we plan to collect comprehensive contextual information for all terms and types within the UMLS and GO datasets. Leveraging this data, we intend to train open-source models using Method P-F (as detailed in Section 3.2.3) and subsequently evaluate their performance. This approach will allow us to assess the impact of enriched context on model accuracy and effectiveness.

Hierarchical Training of Classifiers

We propose a hierarchical training approach for classifiers based on type hierarchies. Specifically, for datasets with multiple levels of types, such as those with 9 top-level categories and corresponding subcategories, we will first train classifiers for the top-level categories and then separately for the subcategories. The performance of this hierarchical approach will be compared to that of a single classifier trained on all types to evaluate the benefits and limitations of each method.

Advancing Prompting

Enhancing prompt engineering strategies for both GPT-3.5-turbo-0125 and Llama-3-8B could lead to improved performance across a wider range of tasks. One potential strategy involves incorporating feedback loops to continuously optimize prompts based on model outputs [29]. This iterative process may include human-in-the-loop systems or automated mechanisms designed to detect and correct errors or inconsistencies in responses. By iteratively refining prompts through continuous optimization, we can enhance overall model performance, ensuring more accurate and reliable outputs.

6 Conclusion

The experiments conducted with GPT-3.5-Turbo, despite not utilizing the latest commercial models such as GPT-4o, yielded promising results. In Task A, GPT-3.5-Turbo outperformed fine-tuned open-source models. However, it's important to note that these commercial models are closed-source, and the cost associated with domain-specific fine-tuning can be prohibitive.

Incorporating domain-specific information and providing a list of classification types within the prompt significantly enhances inference performance. A particularly effective strategy involves initially classifying terms into higher-level categories before refining them into more specific subcategories. This approach not only improves classification accuracy but also reduces the number of tokens required in the prompt.

Our study offers valuable insights into the performance of GPT-3.5-Turbo and Llama-3-8B on complex NLP tasks, underscoring the importance of training methodologies and the role of contextual information in boosting model accuracy. These findings carry helpfulness for the development of future models and methodologies in natural language processing. Notably, the fine-tuned open-source models demonstrated competitive performance on the GeoNames dataset, rivaling that of GPT-3.5-Turbo.

Supplemental Material Statement

Our LLM prompt templates, comprehensive results, and the entire code-base are available as supplementary material on GitHub: https://github.com/MouYongli/LLMs4OL.

Author Contributions

Yixin Peng: Conceptualization, Investigation, Methodology, Data Curation, Formal Analysis, Writing - Original Draft.
Yongli Mou: Supervision, Visualization, Project Administration, Writing - Review & Editing.
Bozhen Zhu: Data Curation.

Sulayman Sowe: Writing - Review & Editing.
Stefan Decker: Funding Acquisition, Supervision.

Competing interests

The authors declare that they have no competing interests.

Acknowledgement

This work was supported by the German Ministry for Research and Education (BMBF) project WestAI (Grant no. 01IS22094D).

References

[1] T. R. Gruber, "Toward principles for the design of ontologies used for knowledge sharing?" *International journal of human-computer studies*, vol. 43, no. 5-6, pp. 907–928, 1995.

[2] C. Biemann, "Ontology learning from text: A survey of methods," *Journal for Language Technology and Computational Linguistics*, vol. 20, no. 2, pp. 75–93, 2005.

[3] 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.

[4] 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.

[5] 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.

[6] S. Gururangan, A. Marasović, S. Swayamdipta, *et al.*, "Don't stop pretraining: Adapt language models to domains and tasks," in *Proceedings of the 58th Annual Meeting of the Association for Computational Linguistics*, D. Jurafsky, J. Chai, N. Schluter, and J. Tetreault, Eds., Online: Association for Computational Linguistics, Jul. 2020, pp. 8342–8360. DOI: 10.18653/v1/2020.acl-main.740. [Online]. Available: https://aclanthology.org/2020.acl-main.740.

[7] Z. Ke, Y. Shao, H. Lin, T. Konishi, G. Kim, and B. Liu, "Continual pre-training of language models," *arXiv preprint arXiv:2302.03241*, 2023.

[8] T. Scialom, T. Chakrabarty, and S. Muresan, "Fine-tuned language models are continual learners," *arXiv preprint arXiv:2205.12393*, 2022.

[9] R. Han, X. Ren, and N. Peng, "Econet: Effective continual pretraining of language models for event temporal reasoning," *arXiv preprint arXiv:2012.15283*, 2020.

[10] T. Wu, L. Luo, Y.-F. Li, S. Pan, T.-T. Vu, and G. Haffari, "Continual learning for large language models: A survey," *arXiv preprint arXiv:2402.01364*, 2024.

[11] L. Wang, X. Zhang, H. Su, and J. Zhu, "A comprehensive survey of continual learning: Theory, method and application," *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 2024.

[12] H. Shi, Z. Xu, H. Wang, *et al.*, "Continual learning of large language models: A comprehensive survey," *arXiv preprint arXiv:2404.16789*, 2024.

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

[14] Y. Lu, X. Zhao, and J. Wang, “Medical knowledge-enhanced prompt learning for diagnosis classification from clinical text,” in *Proceedings of the 5th Clinical Natural Language Processing Workshop*, 2023, pp. 278–288.

[15] J. Liu and L. Yang, “Knowledge-enhanced prompt learning for few-shot text classification,” *Big Data and Cognitive Computing*, vol. 8, no. 4, p. 43, 2024.

[16] 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.

[17] Princeton University, *Wordnet*, Accessed: 2024-07-28, 2024. [Online]. Available: https://wordnet.princeton.edu/.

[18] GeoNames, *Geonames*, Accessed: 2024-07-28, 2024. [Online]. Available: https://www.geonames.org/export/codes.html.

[19] UMLS, *Unified medical language system*, Accessed: 2024-07-28, 2024. [Online]. Available: https://www.nlm.nih.gov/research/umls/index.html.

[20] Gene Ontology, *Gene ontology*, Accessed: 2024-07-28, 2024. [Online]. Available: https://www.geneontology.org/.

[21] Schema.org, *Schema.org*, Accessed: 2024-07-28, 2024. [Online]. Available: https://schema.org/.

[22] MediaWiki API, *Api: Main page - mediawiki*, Accessed: 2024-07-28, 2024. [Online]. Available: https://www.mediawiki.org/wiki/API:Main_page.

[23] OpenAI, *Gpt-4o*, Accessed: 2024-07-28, 2024. [Online]. Available: https://openai.com/index/hello-gpt-4o/.

[24] Anthropic, *Claude 3.5*, Accessed: 2024-07-28, 2024. [Online]. Available: https://www.anthropic.com/claude.

[25] Microsoft, *Copilot: Ai-powered assistance in bing*, Accessed: 2024-07-28, 2024. [Online]. Available: https://copilot.microsoft.com/.

[26] P. Micikevicius, S. Narang, J. Alben, *et al.*, “Mixed precision training,” *arXiv preprint arXiv:1710.03740*, 2017.

[27] I. Loshchilov and F. Hutter, “Decoupled weight decay regularization,” *arXiv preprint arXiv:1711.05101*, 2017.

[28] D. P. Kingma, “Adam: A method for stochastic optimization,” *arXiv preprint arXiv:1412.6980*, 2014.

[29] A. Madaan, N. Tandon, P. Gupta, *et al.*, “Self-refine: Iterative refinement with self-feedback,” *Advances in Neural Information Processing Systems*, vol. 36, 2024.