Building Large Language Models Ontology with Protégé

Ontologies

Authors: Abbas Awarkeh
Hawawou Oumarou Tchapchet
Pin-Xun Huang
Naava Namuwonge Hedwig

Instructed by: Mathieu D'AQUIN
Academic Year 2024-2025

1 Introduction to the LLM Ontology

The Large Language Model Ontology provides a structured framework for understanding key aspects of LLMs. It is designed not only to support researchers and students but also to offer a clear and accessible view for those who may not fully understand LLMs. This ontology covers the architecture, performance metrics, and applications of large language models, giving users a foundational understanding of how LLMs function. It serves as an overview of the LLM domain and a starting point for future ontologies on the subject.

2 Development Process

The ontology was constructed by reviewing existing ontologies, with significant inspiration from the BioPortal Artificial Intelligence Ontology. This iterative process synthesized insights from various sources and advancements in LLM technology.

We used competency questions to guide the development, helping identify essential concepts and relationships.

2.1 Competency Questions

The competency questions we formulated include:

1. What optimization algorithms can be used to train LLMs?
2. What are strategies for mitigating computational resources when training LLMs?
3. What is the environmental impact of LLMs?
4. How many parameters are used for training LLMs?
5. In which domains can LLMs be applied to?
6. What are the architectures used in LLMs?
7. Which evaluation metrics can be used to assess the performance of LLMs?
8. Challenges associated with developing LLMs?

These competency questions helped us refine our ontology by clarifying the key attributes, classes, and relationships needed to address them effectively.

3 Ontology Structure

3.1 Key Classes

- **LLM_Tasks:** Functions like question answering, summarization, and code generation.
- **Architecture:** Design types such as transformer-based models.
- **Training:** Covers methods like supervised learning and optimization techniques.
- **Training_Data:** Types of data used, such as text and multimodal data.

- **Challenges:** Ethical issues, bias, and environmental concerns.
- **Evaluation_Metric:** Human and performance metrics.

3.2 Key Properties

The ontology includes properties like:

- **hasArchitecture:** Links an LLM to its architecture.
- **hasTrainingData:** Associates an LLM with its training data.
- **hasParameters:** Specifies the number of parameters in the LLM.
- **hasChallenge:** Links to challenges like bias or environmental impact.

3.3 Key Individuals

The ontology features notable individuals such as:

- **LLMs:** BERT, GPT-4, T5
- **Training Data:** Common_Crawl, BookCorpus
- **Computational Resources:** GPU, TPU

4 Addressing Competency Questions

The ontology effectively addresses the core competency questions by defining relevant classes, properties, and relationships. It captures key characteristics of LLMs, the role of transformer architectures, the significance of training phases, and factors like training duration, corpus size, and environmental impact.

4.1 Example Competency Questions Answered Using Protégé

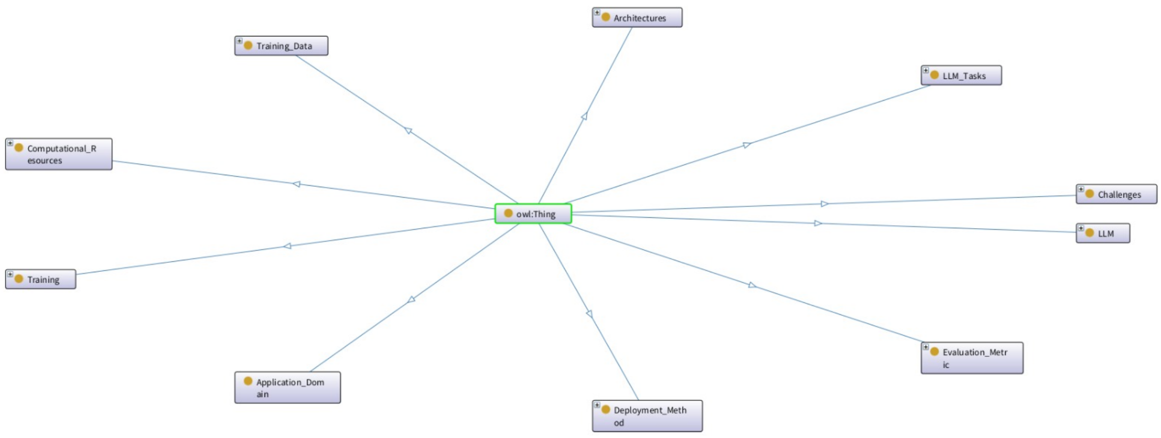
Using Protégé, our ontology can answer the following competency questions:

- **What is the architecture used by GPT-4?**
  By querying the property `hasArchitecture` for the individual GPT-4, Protégé will return the answer "Transformer-based -¿ Decoder only"
- **What training data was used for BERT?**
  Querying the property `hasTrainingData` for the individual BERT will return "Book-Corpus" and "Wikipedia."
- **What challenges are associated with LLMs?**
  By querying the property `hasChallenge`, we can retrieve challenges like Bias and Environmental Impact linked to LLMs.

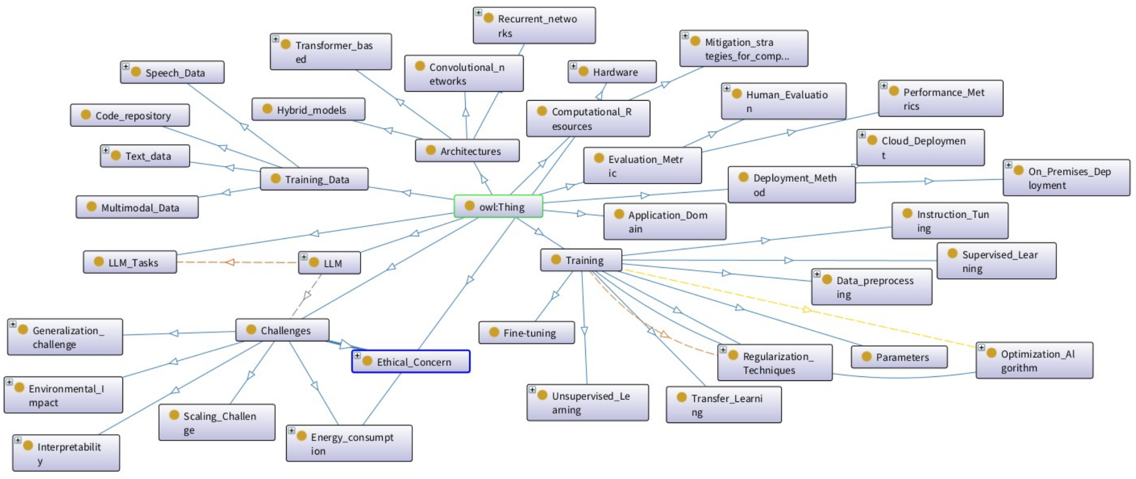
These examples show how ontology can directly answer key competency questions in Protégé by leveraging its classes, properties, and relationships.

5 Ontology Visualization

We examined our ontology and created an OWL document for it. Additionally, we captured a screenshot in Protégé that displays the basic structure, as it is quite extensive. The detailed view of our ontology can be accessed in the file `index-all.html`, which offers a comprehensive overview of the ontology structure and the relationships among its various components.



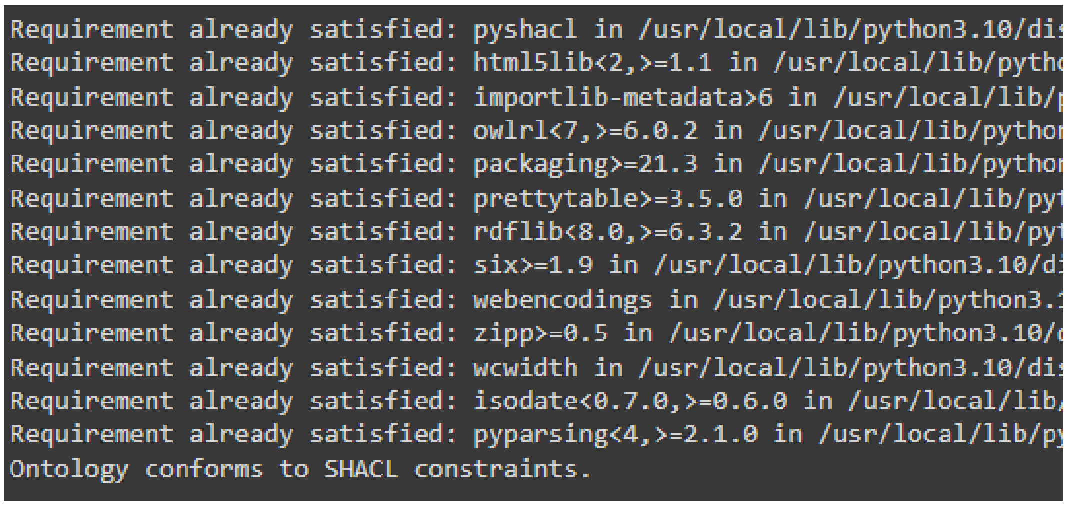
Furthermore, we also have an expanded version of the ontology that provides additional details and insights into its components and relationships.



6 Challenges

We encountered several challenges during the development of our ontology. We faced disjoint errors with our classes, and our ontology continuously displayed an 'OWL nothing' error. After troubleshooting, we discovered that the issue was related to the language tags.

Additionally, we utilized SHACL (Shapes Constraint Language) for validation, and we are pleased to report that our ontology conformed to all SHACL validation requirements, including those checked by the HermiT reasoner.



```
Requirement already satisfied: pyshacl in /usr/local/lib/python3.10/di
Requirement already satisfied: html5lib<2,>=1.1 in /usr/local/lib/pytho
Requirement already satisfied: importlib-metadata>6 in /usr/local/lib/
Requirement already satisfied: owlrl<7,>=6.0.2 in /usr/local/lib/pytho
Requirement already satisfied: packaging>=21.3 in /usr/local/lib/pytho
Requirement already satisfied: prettytable>=3.5.0 in /usr/local/lib/py
Requirement already satisfied: rdflib<8.0,>=6.3.2 in /usr/local/lib/py
Requirement already satisfied: six>=1.9 in /usr/local/lib/python3.10/d
Requirement already satisfied: webencodings in /usr/local/lib/python3.
Requirement already satisfied: zipp>=0.5 in /usr/local/lib/python3.10/
Requirement already satisfied: wcwidth in /usr/local/lib/python3.10/di
Requirement already satisfied: isodate<0.7.0,>=0.6.0 in /usr/local/lib
Requirement already satisfied: pyparsing<4,>=2.1.0 in /usr/local/lib/p
Ontology conforms to SHACL constraints.
```