# PoC: LLM-driven Ontology

The PoC covers:

- **Structured** and **unstructured** illustrative **data** about products and logistics.
- **LLM agents** that **generate ontologies** and **retrieve** data to answer users' questions.

*The PoC presents how I would approach some issues of Ontologies-LLM integration in a (simplified)* ***edge-case scenario****.*

# How to Build Logic Models That Handle Complex Data?

LLM can **generate** consistent **logic models**, also when the *prior knowledge* (e.g., documents and prompts) is **qualitative** and **partial**.

LLM can overcome issues of traditional *expert systems* since hand-made data models often are unreliable and hard to maintain/extend.

**The PoC stresses this aspect by providing limited prior knowledge to LLM prompts.**

*However, when available, more detailed information about the scenario and tasks will likely increase accuracy.*

# How Can LLM Leverage Ontologies to Retrieve Data?

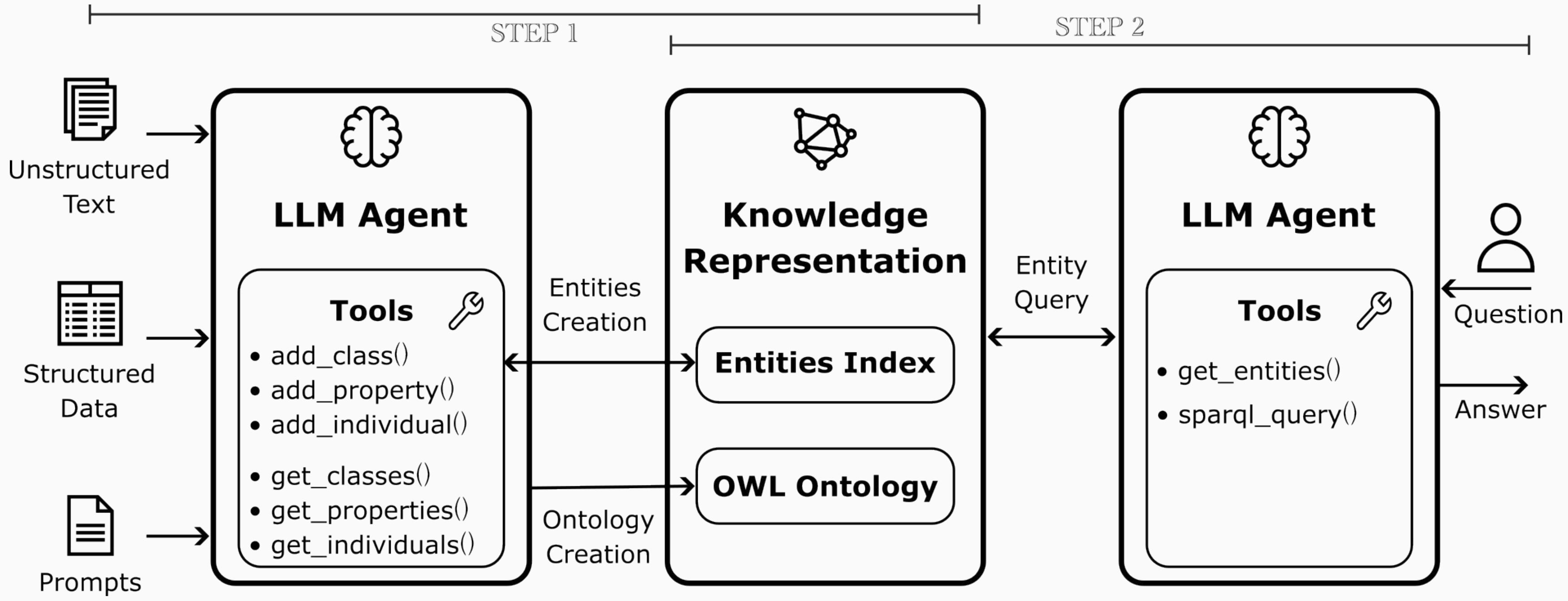
Highly linked data is well-suited for being contextualised in an **ontology** that the **LLM can explore** and benefit from (e.g., via SPARQL queries).

At the cost of *system* complexity, ontologies have several benefits. For example, they support sparse data retrieval, explainability, context-based agent orchestration, consistency checking and error recovery.

**The PoC stresses this aspect by enabling the LLM to retrieve data through SPARQL instead of vector search.**

*However, a combination of vector-search and queries would increase accuracy and decrease computation time.*

# Implementation



STEP 1

STEP 2

Unstructured Text

Structured Data

Prompts

**LLM Agent**

**Tools**

- add_class()
- add_property()
- add_individual()
- get_classes()
- get_properties()
- get_individuals()

Entities Creation

Ontology Creation

**Knowledge Representation**

**Entities Index**

**OWL Ontology**

Entity Query

**LLM Agent**

**Tools**

- get_entities()
- sparql_query()

Question

Answer

# Results

## *Step 1:* **Ontology Creation**

- The LLM represented **structured** data in the ontology **better** than **unstructured** text.
- The LLM **annotated** ontological entities to enhance **reasoning based on natural language**.
- The generated ontology (about 900 facts) had few **redundancies** and was **human-readable** (i.e., suitable for explainability and maintainability).

## *Step 2:* **Data Retrieval**

- The **LLM explored the ontology** and answers were found with **chains of queries**.
- Over **20 questions**: 16 were correct, 1 was partially correct, 1 was incorrect, and 3 had no answer.
- The **LLM explained** the performed queries in a **reproducible** manner.

# Limitations and Improvements

The PoC has a large room for improvement, especially when addressing these limitations:

- The implemented LLM tools address a **fragment of the OWL formalism**, and they do not allow for automatic instance classification.
- **OWL Reasoners** have not been used to support LLM. For example, they could provide **consistency feedbacks** to the LLM for **self-improvement** and **error recovery**.
- LLM with limited prior knowledge generates **granular ontologies**. More **prior knowledge** and a **fine-tuned** LLM would **decrease the number of facts** in the ontology with no semantic loss.
- The PoC does not implement **concurrent LLM agents**, which would decrease computation time.
- Retrieving data only with SPARQL is an extreme case. **Ontologies integrated with Graph or Vector DBs** would improve data retrieval performance.

# Thanks for your attention

More details are available in the PoC repository:
https://github.com/buoncubi/PoC_OWL-LLM_Integration

Luca Buoncompagni
06/11/2025