Leveraging Knowledge Graphs inference for Semi-Explainable Systems based on Large Language Models

Carmelo Fabio Longo[1,*], Daniele Francesco Santamaria[2], Misael Mongiovì[1,2], Luana Bulla[1,2] and Emilio M. Sanfilippo[1]

[1]National Research Council, Institute of Cognitive Science and Technology, Italy
[2]Department of Mathematics and Computer Science, University of Catania, Italy

Abstract
Recent advances in natural language processing concerning Large Language Models (LLMs) raise the challenge of their integration with ontological models, aiming to harness the features of Knowledge Graphs (KG) alongside the expressive abilities of LLMs. This paper introduces QuLIO-XR, a framework designed to integrate these two methods, proposing an approach combining reasoning capabilities of OWL 2 with the expressive power of an LLM. Natural language text is structurally and semantically represented through instances of the foundational ontology called LODO, which combines straightforward notation with human-like reasoning capabilities, addressing issues occurring from the expressive arbitrariness of natural language. Experiments demonstrate also promising translation performance from triples to natural language, establishing QuLIO-XR as a valuable tool in the realm of LLMs explainability, when they are fine-tuned with the same knowledge employed to build LODO KGs.

Keywords
Semantic Web, Large Language Models, Knowledge Graph, Natural Language Processing

1. Introduction

The advent of Large Language Models (LLMs) has ushered in a new era of natural language understanding and generation. These models, such as GPT-3, have demonstrated remarkable capabilities in generating human-like text across a wide range of domains. However, while LLMs excel at natural language processing tasks, their integration with structured knowledge representations, particularly Knowledge Graphs (KG), remains a challenge.

KGs provide a powerful way to organize and represent information in a structured format, offering rich semantic connections between entities and their attributes. On the other hand, LLMs offer unparalleled generative abilities, allowing them to produce coherent and contextually relevant text. The integration of these two technologies holds high potential, combining the structured knowledge representation of KGs with the natural language fluency of LLMs.

14th International Conference on Formal Ontology in Information Systems (FOIS 2024)
*Corresponding author.
fabio.longo@istc.cnr.it (C. F. Longo); daniele.santamaria@unict.it (D. F. Santamaria); misael.mongiovi@istc.cnr.it (M. Mongiovì); luana.bulla@istc.cnr.it (L. Bulla); emilio.sanfilippo@cnr.it (E. M. Sanfilippo)
0000-0002-2536-8659 (C. F. Longo); 0000-0002-4273-652 (D. F. Santamaria); 0000-0003-0528-5490 (M. Mongiovì); 0000-0003-1165-853X (L. Bulla); 0000-0003-2511-2853 (E. M. Sanfilippo)

This paper presents the QuLIO-XR framework, a novel approach to integrate LLMs with KGs representing natural language utterances. By harnessing the reasoning capabilities of OWL 2, QuLIO-XR leverages the features of a foundational ontology defined as LODO [1], which aims to bridge the gap between natural language ontology and human-like fashioned reasoning. The framework permits to populate instances of LODO with RDF triples generated from natural language sentences, by producing also horn-like rules in the SWRL language whom address some issues of the natural language ontology. Furthermore, it enables to query KGs by returning responses in natural language by leveraging the expressive capabilities of LLMs, in order to overcome their known inference limitations. Concerning such limitations, the author of [2] provides a comprehensive analysis of ten categories of ChatGPT's failures, including reasoning, factual errors, math, coding and bias. As a consequence, LLMs employment in real-world scenarios can be critical, especially in sensitive sectors such as healthcare, finance, juridical, and so on, which in this work are being defined as *hot* topics. These considerations, together with the growing interest around such expressive power, motivated the need of studying explainability and interpretability of LLMs. In this context, the QuLIO-XR framework, when used in parallel mode, i.e., by feeding with same data both its integrated LLMs and KGs, can be also a valid tool for explainability of LLMs inferences, since generally LLMs ground truth are not accessible, and even to overcome their inference capabilities but still holding the same expressive power.

The code of QuLIO-XR will be publicly available for research purposes through a dedicated GitHub repository[1], including a RESTful[2] interface to be queried locally or remotely.

The paper is organized as follows. Section 2 provides an overview of the current state-of-the-art in the topic; Section 3 delves into the framework modules; Section 4 offers a comprehensive exploration of our experimental settings, including *Fine-Tuning*, *Evaluation results*, and *Discussion.* Finally, Section 5 concludes the paper with some final considerations.

2. Related works

In this section we report some of the most representative works involving integration between knowledge graphs and LLMs. Such integration can be distinguished in three categories: *KG-enhanced LLMs*, *LLM-augmented KGs* and *Synergized LLMs + KGs*, depending on the starting baseline. As for *KG-enhanced LLMs*, to address the hallucinations issue [3], some researchers [4, 5] proposed to incorporate KGs into LLMs during either pre-training or inference stage, in order to enrich LLMs latent space with knowledge from KGs. Such approaches do not prevent completely allucinations to happen, buy they can constitute a valid starting point for more reliable text generation concerning specific topics. The authors of [6] propose an approach called FLARE (Forward-looking active retrieval augmented generation) to address the pitfalls of traditional Retrieval Augmented generation (RAG) techniques [7] by incorporating feedback from humans and adding more labeled examples in the fine-tuning process. To enhance the comprehensibility of LLMs, scholars also employ KGs to elucidate the facts [8] and the reasoning mechanisms of LLMs [9].

[1] https://github.com/cfabiolongo/qulio-xr

[2] A REST-type (Representational State Transfer) interface for software systems to communicate over the internet using standard HTTP methods and URLs to access resources.

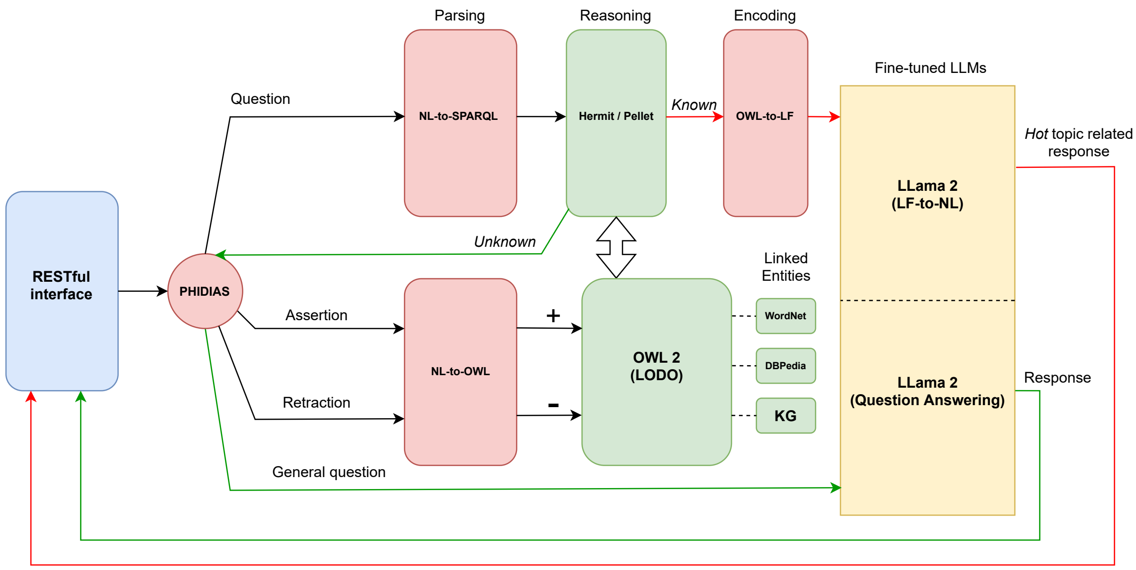

Figure 1: The simplified functional schema of the QuLIO-XR architecture

With respect to *LLM-augmented KGs*, it is intended either when handling incomplete KGs [10] or processing text corpus to construct KGs [11] by leveraging the LLMs generalizability. These studies aim to LLMs prompting design in order to extract relations and entities which are then integrated into KGs, possibly by involving also fine-tuning to focus more generated items to specific topics.

Finally, we intend *Synergized LLMs + KGs* when LLMs and KGs are being unified into a general framework to mutually enhance each other, as the proposal of [12]. This paper's proposed framework can be ascribed to such a category.

3. The Framework

The framework presented in this paper is called ***QuLIO-XR***, standing for ***Qu****erying* ***LI****nguistic* ***O****ntologies with e****X****pressive* ***R****esponse*. The core of this framework processing is managed by the Belief-Desire-Intention [13] framework PHIDIAS [14], which provides routing functions by giving programs the ability to perform logic-based reasoning (in Prolog style) and allows developers to write reactive procedures, i.e., pieces of programs in high level language[3] that can promptly respond to events. The rationale behind such a choice lies in the possibility given by PHIDIAS production rule systems of recombining sequences of beliefs in a compact and straightforward to modify way. For this work, beliefs are pieces of text and labels that are asserted in the PHIDIAS knowledge base, and that can match with one or more production rules and let a specific plan to be executed. The latter can be either (both) further beliefs assertions or (and) the execution of code in a high level language.

[3]The framework is currently available in Python and C++.

Figure 1 shows a simplified functional schema of the framework. First, a sentence is submitted to the **RESTful interface** and sent to **PHIDIAS**, which distinguishes whether a sentence is either assertion/retraction or question. In case of assertion/retraction, the module **NL-to-OWL** translates the sentence in a triples in OWL 2 to be added or removed from the KG. Each entity in such an ontology might be also linked to other KGs in the Semantic Web in order to let them optionally participate in inferences involving the integrated reasoner (Hermit [15] or Pellet [16]). In case of detected question, the text is parsed by the module **NL-to-SPARQL**, translated in SPARQL language, and sent to the reasoner, which attempts a graph matching operation with the LODO ontology in OWL 2. The inferred triples (red branch, i.e. *hot* topic) are translated in logical form by the module **OWL-to-LF** and submitted to an LLM fine-tuned to generate an expressive responses in natural language from logical forms. Otherwise (green branch), assuming that the LODO instance doesn't include such a knowledge (in the open-world assumption), the question in natural language is submitted to another LLM fine-tuned on the Question Answering (QA) task. The two LLMs can be either separated or combined together sharing the space of a single model for their weights.

The following subsections report more details for each of the above modules of the framework.

3.1. The NL-to-OWL Translator

The translation from natural language to KG is achieved starting from text dependencies matching a production rule system, in order to build an intermediate semantic structure defined as *Macro Semantic Table* (MST), which summarizes in a canonical shape all the semantic entities and their relations in a sentence.

Here is a general schema of a MST, as a set of tuples' lists referred to the utterance u:

$$\mathtt{MST(u) = \{ACTIONS,\ VARLIST,\ PREPS,\ BINDS,\ COMPS,\ CONDS\}} \tag{1}$$

where:

- $\mathtt{ACTIONS} = [(\mathtt{label}_k, \mathtt{e}_k, \mathtt{x}_i, \mathtt{x}_j)]$
- $\mathtt{VARLIST} = [(\mathtt{x}_i, \mathtt{label}_i)]$
- $\mathtt{PREPS} = [\emptyset \mid (\mathtt{label}_{p_k}, \mathtt{e}_k, \mathtt{x}_j) \mid (\mathtt{label}_{p_i}, \mathtt{x}_i, \mathtt{x}_j)]$
- $\mathtt{BINDS} = [\emptyset \mid (\mathtt{label}_i, \mathtt{label}_{b_i})]$
- $\mathtt{COMPS} = [\emptyset \mid (\mathtt{label}_j, \mathtt{label}_{c_j})]$
- $\mathtt{CONDS} = [\emptyset \mid \mathtt{e}_k]$

with $k \in \{1, \dots m\}$ and $i, j \in \{1, \dots n\}$.

Briefly, the tuples in `ACTION` represent all verbal interactions inside a sentence related to the verb $\mathtt{label}_k$, plus a reference e_k called *davidsonian*[4] variable and two more variables referencing subject/object; `VARLIST` contains all ground values of variables involved in `ACTIONS`; `PREPS` all preposition involving items from either `ACTIONS` or `VARLIST`; `BINDS` contains all adjectives linked to items from `VARLIST`; `COMPS` contain compound items linked to others items in `VARLIST`;

[4]Inspired by the event-based formal representation due to Davidson [17].

CONDS contains references to items in ACTIONS subordinating the remaining ones, which will be used to assert implicative rules in *Semantic Web Rule Language* (SWRL) [18], an extension of OWL with Horn-like axioms; PREPS, BINDS, COMPS and CONDS can be empty sets (∅). For the sake of shortness, for a detailed overview of MST building the reader can refer to [19]. For instance, considering the following sentence: *When the sun shines, Robert is happy*, the related MST is:

$$\texttt{ACTIONS = [(shine01:VBZ, } e_1\texttt{, } x_1\texttt{, } x_2\texttt{), be01:VBZ(}e_2\texttt{, } x_3\texttt{, } x_4\texttt{)],}$$
$$\texttt{VARLIST = [(}x_1\texttt{, sun01:NN), (}x_2\texttt{, ?), (}x_3\texttt{, Robert01:NNP), (}x_4\texttt{, happy01:JJ)],}$$
$$\texttt{CONDS = [}e_1\texttt{].}$$

where entities inside each tuple are in the shape of "Lemma+Numeration:Part-of-Speech"[5]; "Numeration" prevents from ambiguities in case of words repetition within a sentence, while "Part-of-Speech" (POS)[6] is necessary for the subsequent operations involving LLM fine-tuning/inference. Furthermore, the question mark coupled with x_2 indicates that "shine" has no object, therefore in this case it is considered intransitive verb.

Starting from such MST structure, the NL-to-OWL translator builds, through a production rule system, an ontological representation directly related with the linguistic features of sentences. Furthermore, the ontology includes also a set of domain-specific rules, suitably constructed with SWRL axioms, that provide the ontologies with additional reasoning features in a human-like fashion. Each ontology built with such criteria belongs to a specific family, which for its direct derivation from the Davidson notation we define as **LODO** ontology (**L**inguistic **O**riented **D**avidsonian **O**ntology). LODO can be considered as a foundational ontology, i.e., a specific type of ontology designed to model high-level and domain-independent categories about the real world.

The general schema of LODO is quite straightforward: we define *regular*[7] *verbal phrase* by means of the following OWL classes:

- ***Entity***. Instances of this class represent entities referenced by either the object-property *hasSubject* or *hasObject*. Compound nouns are concatenated in order to form a single individual.
- ***Verb***. Each instance of this OWL class represents a verbal phrase inspired from the Davidsonian notation. They exploit the following object-properties: *hasId*, whose value is a unique identification code; *hasSubj*, which represents the verb subject in the domain of *Entity*; *hasObj* which represents the verb object in the domain of either *Entity* or *Verb* (in the case of embedded verbal actions); *isPassive* (optional), indicating whether a verbal action is passive or not. A typical instance *Verb* involves the following entities and triples, with *Subj* and *Obj* instances of the class *Entity* as follows:

Verb(x1), hasSubj(x1, x2), Subj(x2), hasObj(x1, x3), Obj(x3).

[5] Assuming + as strings concatenation operator.
[6] https://github.com/clir/clearnlp-guidelines
[7] The usage of *regular* will be clarified ahead. Biefly, it is employed to distinguish verbal phrases from those that are directly translated into implicative SWRL axioms.

- ***Id***. Instances of this OWL class represent unique identification codes (e.g. a timestamp) related with verbal actions. These individuals are the ones involved as value of the object property *hasId*, which is used with instances of the class *Verb*. Instances of the class *Id* can be useful to deal with inconsistency cases: the higher is the *Id*, the more valid is the related instance of *Verb*, even when the latter has the property *hasAdverb* expressing the value *Not*;[8] Furthermore, the object property *hasTime* and *hasPlace* can be used to express the times and places possibly inferred from Named Entity Recognition (NER). A typical instance is used as follows:

  Verb(x1), hasId(x1, x2), Id(x2).

- ***Adjective***. Instances of this class take the values of the object-property *hasAdj* together with instances of the class *Entity* (*Subj*) as follows:

  Subj(x1), hasAdj(x1, x2), Adj(x2).

- ***Preposition***. Instances of this class represent prepositions and are referenced by the object-property *hasPrep* with instances of either the class *Verb* or the class *Entity*. Moreover, instances of *Preposition* (*Prep*) are involved in the object property *hasObject* referencing instances of the class *Entity* (*Subj* and *Obj*). For example, instances of *Preposition* are used as follows:

  Subj(x1), hasPrep(x1, x2), Prep(x2), hasObj(x2, x3), Obj(x3),

  or

  Verb(x1), hasPrep(x1, x2), Prep(x2), hasObj(x2, x3), Obj(x3).

  As in natural language, such properties can represent also linkage of verbs/entities to times and places, although natural language preposition as *at*, *on* and *in* can be linked to either times or space references; thus proper disambiguation criteria must be considered to avoid ambiguities in the knowledge representation.

- ***Adverb***. Instances of this class represent adverbs and have the values of the object property *hasAdv* together with instances of the class *Verb*:

  Verb(x1), hasAdv(x1, x2), Adv(x2).

In addition to the classes described above, LODO comprises also a group of default SWRL rules implicitly created by QuLIO-XR with the aim of increasing the reasoning capabilities of the applications. Such rules are classified as follows:

- ***Assignment Rules***. These rules are implicitly asserted in the presence of a copular[9] verb for the ROOT dependency. Formally, in the presence of the following tuples in MST:

[8] Negations are treated as whatever adverbs, although their employment depends on the domain. In linguistic science, intentional objects as *nonexistent* are considered particularly problematic [20].

[9] A copular verb, also called a linking verb, connects the subject of a sentence to a subject complement, usually an adjective or a noun, providing more information about the subject without showing action. Examples include "be," "seem," and "appear."

```
ACTIONS = [(Cop:POS, e1, x1, x2)]
VARLIST = [(x1, Subject:POS), (x2, Object:POS))]
```

where each predicate has its own POS tag. Such a expression triggers the following SWRL rule, by omitting the POS for the sake of shorteness:

$$\textit{Subject(?x) -> Object(?x).} \tag{2}$$

The rationale of such a rule is that, by the virtue of the copular verb, the class membership of the verb's object is inherited by the subject.

- ***Legacy Rules***. Legacy rules are implicitly asserted together with the *Assignment Rules*, to allow a copular verb's subject to inherit both adjectives and preposition properties of the verb's object. The following legacy rule[10] is built together with (2):

  Subject(?x1), Object(?x2), hasAdj(?x1, ?x3), Adjective(?x3) -> hasAdj(?x2, ?x3).

- ***Deadjectival Rules***. In presence of an instance of *Adjective*, deadjectival rules assert new *deadjectivated* instances of the class *Adjective* as new memberships of the adjective related noun, in order to improve reasoning. A deadjectival rule has the following form:

  Entity(?x1), hasAdj(?x1, ?x2), Adjective(?x2) -> Entity(?x2).

- ***Deverbal Rules***. In the presence of an instance of *Verb*, such rules assert new *deverbalized* instances of the class *Verb* in order to improve reasoning. By leveraging lexical resources as WordNet[21] it is possible to infer whether a word can have, depending on the sentence semantic, either noun of verbal role. In this way, for instance, the sentence: *I have a walk* has the same meaning of *I walk*, therefore both versions of sentences may coexist in the same KG to increase reasoning capabilities.
- ***Implicative Copular Rules***. These rules are built from CONDS content when a subordinating verbal action's (`Verb:POS`) subject is referred to the same entity of the subject of a copular verb (`Cop:POS`). Such rules are useful to infer new memberships of the initial sentence subject (which is required to be present also in the body). The MST required to build implicative copular rules must have the following shape:

  ```
  ACTIONS = [(Verb:POS, e1, x1, x2)], (Cop:POS, e2, x3, x4),...],
  VARLIST = [(x1, Subject1:POS),...(x3, Subject1:POS), (x2, Object2:POS),...]
  CONDS = [e1],
  ```

  where corresponding variables for `Subject`$_1$`:POS` in `VARLIST` are in both tuples in `ACTIONS`, which permit the formal assertion of the following pattern:

  $$Subject_1(?x_1), \ldots \text{ -> } Object_2(?x_1).$$

  Such translation is applied, for instance, in the presence of the following sentence: *When Robert drinks wine, Robert is happy*, in order to infer the *happy* membership (i.e. belonging to a group of *happy* people) for *Robert* whenever *Robert drinks wine*.

[10]An analogous rule is generated for preposition, where *hasAdj* is replaced with *hasPrep*.

- ***Value Giver Statements***. These optional rules contribute to assign values to the data property *hasValue* related with the specified individuals, by matching for instance the following tuples in MST (among the possible cases expressing quantification):

$$\texttt{PREPS = [(}x_1\texttt{, "To", }x_2\texttt{)]}$$
$$\texttt{VARLIST = [(}x_1\texttt{, Subject}_1\texttt{), (}x_1\texttt{, "Equal"), (}x_2\texttt{, VALUE),...]}$$

  VALUE specifies the value that must be given to the individual corresponding with the variable x_1 (Subject$_1$). The property *hasValue* might be involved in comparison operations in the writing of SWRL rules.
- ***Values Comparison Conditionals***. These optional rules are parsed from sentences in an analogous way as to the *Value Giver Statement*, but they take place within the body of *Implicative Copular Rules*.

For more details about the LODO foundational ontology and its applications, the reader is referred to [1].

3.2. The NL-to-SPARQL Translator

Similarly to the information encoding in OWL 2 reported in the previous section, MSTs play a crucial role also for Natural Language (NL) to SPARQL translation. Here's a list of possible competency *Polar* and *Wh*-questions, which are translated in SPARQL by leveraging a production rule system matching MSTs.

- ***Polar* assertive questions**. Questions whose response is expected to be either *True* or *False* are the more straightforward to translate from natural language to SPARQL, because any question is translated directly *as is* in a graph matching operation, where an instance of (1) is used to build the body of a query[11]. A notation leveraging the above seen *assignment rules* is expected in the ontology, which permits (in presence of copular verbs as *Be*) a direct membership check of individuals in the SPARQL query.
  – Example: *Colonel West is a criminal?*

$$\texttt{ACTIONS = [(Be:VBZ, }e_1\texttt{, }x_1\texttt{, }x_2\texttt{)]}$$
$$\texttt{VARLIST = [(}x_1\texttt{, Colonel), (}x_2\texttt{, Criminal)]}$$
$$\texttt{COMPS = [(Colonel, West)]}$$

```
PREFIX lodo: <http://test.org/west.owl#>
PREFIX rdf: <http://www.w3.org/1999/02/22-rdf-syntax-ns#>

ASK WHERE {
?c rdf:type lodo:Colonel_West.
?c rdf:type lodo:Criminal.
}
```

  – Example: *Colonel West sells missiles to Nono?*

$$\texttt{ACTIONS = [(Sell:VBZ, }e_1\texttt{, }x_1\texttt{, }x_2\texttt{)]}$$
$$\texttt{VARLIST = [(}x_1\texttt{, West), (}x_2\texttt{, Missiles), (}x_5\texttt{, Nono)]}$$
$$\texttt{PREPS = [(To, }e_1\texttt{, }x_5\texttt{)]}$$
$$\texttt{COMPS = [(Colonel, West)]}$$

[11]We reported POS only on Verbs inside ACTIONS in (1) and for *Wh-* entities.

```
PREFIX lodo: <http://test.org/west.owl#>
PREFIX rdf: <http://www.w3.org/1999/02/22-rdf-syntax-ns#>

ASK WHERE {
?e1 rdf:type lodo:Sells.
?e1 lodo:hasSubj ?x1.
?x1 rdf:type lodo:Colonel_West.
?e1 lodo:hasObj ?x2.
?x2 rdf:type lodo:Missiles.
?e1 lodo:hasPrep ?e1p1.
?e1p1 lodo:hasObj ?x5.
?x5 rdf:type lodo:Nono.
}
```

- ***Who* questions**. The MST of such type of questions contains the entity `WHO:WP` which is identified as the query's *target* before SELECT. The remaining entities from the other MST list are used as in the prior case to populate the *WHERE* section.
  – Example: *Who is Colonel West?*

$$\begin{aligned} \texttt{ACTIONS} &= [(\texttt{Be:VBZ}, e_1, x_1, x_2)] \\ \texttt{VARLIST} &= [(x_1, \texttt{West}), (x_2, \texttt{Who:WP})] \\ \texttt{COMPS} &= [(\texttt{Colonel}, \texttt{West})] \end{aligned}$$

```
PREFIX lodo: <http://test.org/west.owl#>
PREFIX rdf: <http://www.w3.org/1999/02/22-rdf-syntax-ns#>

SELECT ?Who WHERE {
lodo:Colonel_West rdf:type ?Who.
}
```

- ***What* questions**. As in the previous case, the corresponding MST contains the entity `What:WP` which is identified as the query's *target* before SELECT. The remaining entities from the other MST list are used as in the prior case to populate the *WHERE* section.
  – Example: *What does Colonel West sell?*

$$\begin{aligned} \texttt{ACTIONS} &= [(\texttt{Sell:VBZ}, e_1, x_1, x_2)] \\ \texttt{VARLIST} &= [(x_1, \texttt{West}), (x_2, \texttt{What:WP})] \\ \texttt{COMPS} &= [(\texttt{Colonel}, \texttt{West})] \end{aligned}$$

```
PREFIX lodo: <http://test.org/west.owl#>
PREFIX rdf: <http://www.w3.org/1999/02/22-rdf-syntax-ns#>

SELECT ?What WHERE {
?e1 rdf:type lodo:Sells.
?e1 lodo:hasSubj lodo:Colonel_West
?e1 lodo:hasObj ?What
}
```

- ***Where* questions**. In this case, the corresponding MST contains the entity `Where:WRB`[12] linked to the verb inside ACTIONS as the query's *target* before *SELECT*. The remaining entities from the other MST lists are used to populate the *WHERE* section, but the query takes into account also of possible places detected as NER in the assertion, plus possible places given by prepositions.

[12]The POS *WRB* identifies an adverb.

– Example: *Where does Colonel West live?*

$$\texttt{ACTIONS} = [(\texttt{live:VBZ}, e_1, x_1, x_2)]$$
$$\texttt{VARLIST} = [(x_1, \texttt{West}), (x_2, ?), (e_1, \texttt{Where:WRB})]$$
$$\texttt{COMPS} = [(\texttt{Colonel}, \texttt{West})]$$

```
PREFIX lodo: <http://test.org/west.owl#>
PREFIX rdf: <http://www.w3.org/1999/02/22-rdf-syntax-ns#>

SELECT ?Where WHERE
{
?e1 rdf:type lodo:lives.
?e1 lodo:hasSubj lodo:Colonel_West.
?e1 lodo:hasPlace ?Where
}
UNION
{
?e1 rdf:type lodo:lives.
?e1 lodo:hasSubj lodo:Colonel_West.
?e1 lodo:hasPrep ?e1p1
?e1p1 lodo:hasObj ?Where
}
```

- ***When* questions**. Similarly as above, the corresponding MST contains the entity `When:WRB` linked to the verb inside ACTIONS, as the query's *target* before *SELECT*. The remaining entities from the other MST lists are used to populate the *WHERE* section, but the query takes into account also of possible times detected as NER in the assertion, plus possible times given by prepositions.

  – Example: *When was Colonel West born?*

$$\texttt{ACTIONS} = [(\texttt{Born:VBN}, e_1, x_1, x_2)]$$
$$\texttt{VARLIST} = [(x_1, ?), (x_2, \texttt{West}), (e_1, \texttt{When:WRB})]$$
$$\texttt{COMPS} = [(\texttt{Colonel}, \texttt{West})]$$

```
PREFIX lodo: <http://test.org/west.owl#>
PREFIX rdf: <http://www.w3.org/1999/02/22-rdf-syntax-ns#>

SELECT ?When WHERE
{
?e1 rdf:type lodo:Born.
?e1 lodo:hasObj lodo:Colonel_West.
?e1 lodo:hasTime ?When
}
UNION
{
?e1 rdf:type lodo:Lives.
?e1 lodo:hasObj lodo:Colonel_West.
?e1 lodo:hasPrep ?e1p1
?e1p1 lodo:hasObj ?When
}
```

3.3. The OWL-to-LF Translator

This module (crf. Figure 1) is designed to encode triples from LODO KGs into *nested* grounded logical forms , by leveraging a production rule system. Such logical forms comply with the following criteria for snippet and verbal phrases:

- **Snippet.** By supposing the presence of both preposition and adjective in a sentence, given the following preposition related triples:

  (*Entity*, hasPrep, ObjectPrep), (hasPrep, rdf:type, ClassPrep), (ObjectPrep, rdf:type, ClassObjectPrep)

  and the following adjective related triples:

  (*Entity*, hasAdj, ObjectAdj), (hasAdj, rdf:type, ClassAdj)

  with *Entity* referred to the same individual. The translated notation is the following:

  ```
  ClassPrep(ClassAdj(Entity), ClassObjectPrep))
  ```

  otherwise, in case ClassAdj is linked to ClassObjectPrep:

  ```
  ClassPrep(Entity, ClassAdj(ClassObjectPrep))
  ```

  For instance, concerning the snippet: *Protector of the sacred grove*, the corresponding logical form will be enconded as:

  ```
  Of_IN(Protector_NN, sacred_ADJ(Grove_NN)),
  ```

  where IN, NN and ADJ are POS tags included in the notation.
- **Verbal phrase.** In regard of verbal phrase, similarly as above[13] and including also adverbs, one of the possible nesting hierarchy is the following (by supposing the presence of verb, preposition and adverb in a sentence):

  ```
  ClassAdv(ClassPrep(ClassVerb(ClassSubj, ClassObj), ClassObjectPrep))
  ```

  For instance, the corresponding logical form of the sentence: *The mysterious fog hardly enveloped the old graveyard* is:

  ```
  Hardly_WRB(Enveloped_VBD(Mysterious_JJ(Fog_NN), Old_JJ(Graveyard_NN))).
  ```

4. Validation

This section is about validation focused on the performance of two instances of LLama-2-7B-chat, each fine-tuned on a distinct downstream tasks, evaluated in single and combined[14] configuration. Such tasks are the Logical Form to Natural Language (LF-to-NL) translation and Question Answering (QA). In the next subsections we report details about a case-study *multitask* fine-tuning and its performance scores for two datasets, then further subsections about results evaluation and discussion.

[13] I this case the preposition `ClassPrep` is linked to a Verb instead of a noun.
[14] By sharing the same model's latent space for their weights.

4.1. Fine-tuning

In order to endow the framework with LF-to-NL translation and to deal with the QA task, we fine-tuned two distinct instance of LLama2-7B-chat (as shown in Fig. 1), then we combined them in a single multitask model. The LF-to-NL model was fine-tuned with 900 couples: (LF expression, NL sentence)[15], with the following prompt:

Use the Input below to create a sentence in expressive English, which could have been used to generate the Input logical form.

For the above task, ChatGPT 3.5 has been employed to generate a training dataset made of sentences with different length and semantic complexity, in order to deal with more levels of *nested* information, plus snippets (non-verbal phrases) whom may be possible results of logical inference. As logical forms, we used the notation reported in Section 3.3. For instance:

Colonel West sells missiles to Nono,

is represented by the logical form made of the following composite literal:

`To_IN(Sells_VBZ(Colonel_NNP_West_NNP, Missiles_NNS), Nono_NNP).`

The standardized shape of such logical forms, endowed with labels encompassing POS spanning in a set of 36 items[16] (which is enormously less than all the possible words an LLM was trained on), helps greatly LLMs in handling recurrent patterns. Furthermore, since the employed instance of LODO is endowed also to capture data from NER, in presence of *hasPlace* and *hasTime*, the label PLACE and TIME will be used in the encoding together with individual's labels and POS as one of the following literals: `TIME(X, NER(`*time*`))`, `PLACE(X, NER(`*place*`))`, `PLACE(TIME(X, NER(`*time*`), NER(`*place*`))`, where `X` is a logical form corresponding to a verbal phrase and *time/place* as NER values. Concerning the QA fine-tuning, which is possibly related to *hot* topics, the work in [22] suggests that comparable performance can be obtained by constructing a high-quality and low-quantity dataset (around ∼1000 samples) rather than using datasets with lower quality but higher quantity. With that in mind, we employed 1000 *open QA* items from *dolly* [17], an open-source dataset comprising instruction-following records created by numerous Databricks employees. The dataset spans across various behavioral categories outlined in the InstructGPT [23] paper, including brainstorming, classification, open/closed QA, generation, information extraction and summarization. The prompt used for QA fine-tuning is the following:

Generate a response to the question given in Input.

Both fine-tunings have been carried out by using Low-Rank Adaptation (LoRA) [24], which is a Parameter-Efficient Fine-Tuning (PEFT) method that decomposes a large matrix into two smaller low-rank matrices in the attention layers. This drastically reduces the number of parameters that need to be fine-tuned, turning it into an acceptable option in case of modest computational resources.

[15]The dataset is provided in this work's GitHub repository.
[16]https://www.ling.upenn.edu/courses/Fall_2003/ling001/penn_treebank_pos.html
[17]https://huggingface.co/datasets/databricks/databricks-dolly-15k

Model	MATCH	ADM	AVG Precision/Recall/F1
LF-to-NL	38%	97%	0.976 / 0.979 / 0.977
LF-to-NL (merged)	39%	94%	0.972 / 0.977 / 0.974

Table 1
LF-to-NL translation scores of LLama2-7B-chat fine-tuned with logical forms, tested on a sample of 100 unseen logical forms (temperature=0.1).

4.2. Evaluation results

The evaluation takes into account three metrics: morphological match (MATCH), admissibility (ADM) by human judgment, and average (AVG) BERT-score[18] (Precision, Recall, and F1), all of them compared with ground truth values from a test dataset. Concerning admissibility, we considered *admissible* those paraphrases of ground truth values sharing the same verbal phrase structure and general meaning. Given *Admissible* a set of detected paraphrases on a sample of inferences and *Groudtruth* the corresponding set of ground truth values, we assume always that *Groudtruth* $\subseteq$ *Admissible*, i.e., any sentence is also a paraphrase of itself. Each ADM reported in this paper is achieved from the average judgment of three distinct annotators.
As for BERT-Score, the work in [25] showed that it correlates well with human judgment on sentence-level and system-level evaluation. For instance, comparing the following ground truth from this work's case-study:

Happily, the birds chirped in the early morning sun

and of the following generated admissible paraphrase:

Birds chirped happily in the early morning sun,

their BERT-score are: 0.930, 0.975, 0.948.

The evaluation was carried out by considering two Llama2-7B-chat instances fine-tuned on distinct downstream tasks: LF-to-NL (Table 1) and QA (Table 2), taking into account a sample of 100 questions from the training dataset and 100 unseen logical forms; we tested the two models for both weights built on each task and also merged them with the base model weights (enabling access to the implicit knowledge of pre-trained LLama-2-chat). Subsequently, we conducted the same tests (Table 3) on the *Multitask* model, endowed of weights achieved from LoRA concatenation of the two adapters[19] related with each task. Concatenation weights are generally domain and task dependent; in this work we choose them[20] to maximize the LF-to-NL performance scores.

4.3. Discussion

A comprehensive experimental analysis was conducted to assess the efficacy of this work. Specifically, we examined the capability of the fine-tuned LLama2-7B-chat model to translate

[18]https://huggingface.co/spaces/evaluate-metric/bertscore
[19]For *adapter* is meant the set of all weights added to the base model during the LoRa fine-tuning operations.
[20]Concatenation weights chosen for LF-to-NL and QA are respectively: [0.7, 0.1].

Model	MATCH	ADM	AVG Precision/Recall/F1
QA	30%	65%	0.922 / 0.913 / 0.917
QA (merged)	46%	76%	0.903 / 0.889/ 0.895

Table 2
QA predictions scores for LLama2-7B-chat fine-tuned with the dolly dataset, tested on a sample test of 100 open QA questions (temperature=0.1).

Model	Prediction	Temp	MATCH	ADM	AVG Precision/Recall/F1
Multitask	LF-to-NL	0.6	34%	92%	0.963 / 0.973 / 0.968
Multitask	QA	0.1	6%	60%	0.778 / 0.827 / 0.802

Table 3
Both LF-to-NL and QA predictions scores, for the Multitask fine-tuned Llama2-7B-chat obtained from the two adapters of Tables 1 and 2, tested on 100 unseen logical forms and 100 open QA questions.

logical forms into natural language for both single adapter and combined with a further adapter related with the QA downstream task. Specifically, Table 1 illustrates the model's performance on the single downstream task of LF-to-NL: on a sample of 100 logical forms, 38% (39% in case of merged weights with the base model) of predictions closely matched the ground truth values (MATCH), while the admissible (ADM) percentage is 97% (94% in case of merged weights with the base model), which is reflected in BERT-scores.

Table 2 shows the scores related to the QA task, revealing that both MATCH add ADM are higher for the merged adapter with a percentage gain of respectively 16% and 11%. The second row shows lower BERT-scores in spite of better MATCH and ADM: the rationale is that by merging the adapter with the base model gives the inference the access to Llama pre-training knowledge, getting the valid responses spectrum wider although semantically distant from ground truths.

Table 3 reveals that LF-to-NL capability is mostly held for the *Multitask* model, for both MATCH (34% vs 38%) and ADM (92% vs 97%) compared with single adapter (not merged, first row in table 1), whom are reflected also in BERT-scores. As for QA task performance conducted on the same tests of Table 2, by comparing results with the first row which has the best BERT-scores, the loss is higher for both MATCH (6% vs 30%) and ADM (60% vs 65%) than by using single adapters. However, a 5% ADM percentage loss for the QA task is an acceptable compromise considering that LF-to-NL capabilities are partially held for the *Multitask* model, compared with single adapters in Table 1, and considering also that both fine-tunings were not tailored to any specific *hot* topic. In any case, we expect better results by employing bigger models than Llama-7B-chat, whose low hardware requirements allowed the prototype to be tested on non-high profile machines.

5. Conclusions

The current work presents an innovative approach to integrate KGs and LLMs, aiming to exploit the advantages of both in terms of reliability, reasoning and expressiveness. In this approach, OWL inference is implicitly activated when questions fall within one or more related *hot* topics, by leveraging an LLM to provide expressive feedback to the user, while the LLM shoulders the entire task for questions unrelated to *hot* topics. Such integration is achieved with a framework called QuLIO-XR, which leverages all features and inference capabilities of a foundational ontology designed for linguistic called LODO. Evaluation of this approach revealed that Llama-2-chat can be fine-tuned for the LF-to-NL downstream task, with logical form directly achieved from KGs complying with LODO, showing promising performance for both single and combined adapters. Since LLMs ground truth are generally not accessible, QuLIO-XR can also serve as evaluation tool for LLMs' predictions, by exploiting its dual OWL/LLM inference mode (which simultaneously presents results from both inference systems) in scenarios of parallel knowledge input for both KGs and LLM fine-tuning.

References

[1] C. F. Longo, C. Santoro, M. Nicolosi-Asmundo, D. Cantone, D. F. Santamaria, Towards ontological interoperability of cognitive iot agents based on natural language processing, Intelligenza Artificiale 16 (2022) 93–112. doi:10.3233/IA-210125.

[2] A. Borji, A categorical archive of chatgpt failures, 2023. URL: https://arxiv.org/abs/2302.03494. doi:10.48550/ARXIV.2302.03494.

[3] Z. Ji, N. Lee, R. Frieske, T. Yu, D. Su, Y. Xu, E. Ishii, Y. J. Bang, A. Madotto, P. Fung, Survey of hallucination in natural language generation, ACM Computing Surveys 55 (2023) 1–38. URL: http://dx.doi.org/10.1145/3571730. doi:10.1145/3571730.

[4] Z. Zhang, X. Han, Z. Liu, X. Jiang, M. Sun, Q. Liu, ERNIE: Enhanced language representation with informative entities, in: A. Korhonen, D. Traum, L. Màrquez (Eds.), Proceedings of the 57th Annual Meeting of the Association for Computational Linguistics, Association for Computational Linguistics, Florence, Italy, 2019, pp. 1441–1451. URL: https://aclanthology.org/P19-1139. doi:10.18653/v1/P19-1139.

[5] C. Rosset, C. Xiong, M. Phan, X. Song, P. Bennett, S. Tiwary, Knowledge-aware language model pretraining, 2021. arXiv:2007.00655.

[6] L. Zhang, K. Jijo, S. Setty, E. Chung, F. Javid, N. Vidra, T. Clifford, Enhancing large language model performance to answer questions and extract information more accurately, 2024. arXiv:2402.01722.

[7] Y. Gao, Y. Xiong, X. Gao, K. Jia, J. Pan, Y. Bi, Y. Dai, J. Sun, Q. Guo, M. Wang, H. Wang, Retrieval-augmented generation for large language models: A survey, 2024. arXiv:2312.10997.

[8] F. Petroni, T. Rocktäschel, P. Lewis, A. Bakhtin, Y. Wu, A. H. Miller, S. Riedel, Language models as knowledge bases?, 2019. arXiv:1909.01066.

[9] B. Y. Lin, X. Chen, J. Chen, X. Ren, KagNet: Knowledge-aware graph networks for commonsense reasoning, in: K. Inui, J. Jiang, V. Ng, X. Wan (Eds.), Proceedings of

the 2019 Conference on Empirical Methods in Natural Language Processing and the 9th International Joint Conference on Natural Language Processing (EMNLP-IJCNLP), Association for Computational Linguistics, Hong Kong, China, 2019, pp. 2829–2839. URL: https://aclanthology.org/D19-1282. doi:10.18653/v1/D19-1282.
[10] Z. Zhang, X. Liu, Y. Zhang, Q. Su, X. Sun, B. He, Pretrain-KGE: Learning knowledge representation from pretrained language models, in: T. Cohn, Y. He, Y. Liu (Eds.), Findings of the Association for Computational Linguistics: EMNLP 2020, Association for Computational Linguistics, Online, 2020, pp. 259–266. URL: https://aclanthology.org/2020.findings-emnlp.25. doi:10.18653/v1/2020.findings-emnlp.25.
[11] A. Kumar, A. Pandey, R. Gadia, M. Mishra, Building knowledge graph using pre-trained language model for learning entity-aware relationships, 2020 IEEE International Conference on Computing, Power and Communication Technologies (GUCON) (2020) 310–315. URL: https://api.semanticscholar.org/CorpusID:226266162.
[12] S. Pan, L. Luo, Y. Wang, C. Chen, J. Wang, X. Wu, Unifying large language models and knowledge graphs: A roadmap, IEEE Transactions on Knowledge and Data Engineering (2024) 1–20. URL: http://dx.doi.org/10.1109/TKDE.2024.3352100. doi:10.1109/tkde.2024.3352100.
[13] A. Rao, M. Georgeff, BDI agents: From theory to practice, in: Proceedings of the first international conference on multi-agent systems (ICMAS-95), San Francisco, CA, 1995, pp. 312–319.
[14] F. D'Urso, C. F. Longo, C. Santoro, Programming intelligent iot systems with a python-based declarative tool, in: The Workshops of the 18th International Conference of the Italian Association for Artificial Intelligence, 2019.
[15] B. Glimm, I. Horrocks, B. Motik, G. Stoilos, Z. Wang, HermiT: An OWL 2 Reasoner, Journal of Automated Reasoning 53 (2014) 245–269.
[16] E. Sirin, B. Parsia, B. C. Grau, A. Kalyanpur, Y. Katz, Pellet: A practical OWL-DL reasoner, Web Semantics 5 (2007) 51–53.
[17] D. Davidson, The logical form of action sentences, in: The logic of decision and action, University of Pittsburg Press, 1967, p. 81–95.
[18] World Wide Web Consortium, SWRL: A Semantic Web Rule Language Combining OWL and RuleML, 2004. URL: http://www.w3.org/Submission/SWRL/.
[19] C. F. Longo, F. Longo, C. Santoro, Caspar: Towards decision making helpers agents for iot, based on natural language and first order logic reasoning, Engineering Applications of Artificial Intelligence 104 (2021) 104269. URL: https://www.sciencedirect.com/science/article/pii/S0952197621001160. doi:https://doi.org/10.1016/j.engappai.2021.104269.
[20] P. van Inwagen, Ontology, Identity, and Modality: Essays in Metaphysics, Cambridge University Press, 2001.
[21] G. A. Miller, Wordnet: A lexical database for english, in: Communications of the ACM Vol. 38, No. 11: 39-41, 1995.
[22] C. Zhou, P. Liu, P. Xu, S. Iyer, J. Sun, Y. Mao, X. Ma, A. Efrat, P. Yu, L. Yu, S. Zhang, G. Ghosh, M. Lewis, L. Zettlemoyer, O. Levy, Lima: Less is more for alignment, 2023. arXiv:2305.11206.
[23] J. Li, T. Tang, W. X. Zhao, J.-R. Wen, Pretrained language models for text generation: A survey, 2021. arXiv:2105.10311.

[24] E. J. Hu, Y. Shen, P. Wallis, Z. Allen-Zhu, Y. Li, S. Wang, L. Wang, W. Chen, Lora: Low-rank adaptation of large language models, 2021. arXiv:2106.09685.
[25] T. Zhang*, V. Kishore*, F. Wu*, K. Q. Weinberger, Y. Artzi, Bertscore: Evaluating text generation with bert, in: International Conference on Learning Representations, 2020. URL: https://openreview.net/forum?id=SkeHuCVFDr.