Actively Learning: an LLM extension for the Exact Learner framework

Riccardo Squarcialupi
riccard.squarcialupi@studio.unibo.it

Introduction I

Large language models (LLMs) have become sophisticated in question-answering, ranging from general knowledge to domain-specific queries.

Recent efforts automate knowledge extraction from LLMs in the form of ontologies.

Ontologies & DL

Ontologies represent conceptual knowledge and are used for data integration, information retrieval, and automated reasoning.

Description logic formalizes ontologies, enabling representation of hierarchical structures.

Active Learning

Active learning is a method developed by Angluin (1987) and involves a learner posing questions to a teacher (syntetic & omniscient) to acquire knowledge.

In computational learning theory, these questions are called **membership queries**, answered with ('true' or 'false').

What if the teacher is an LLM and doesn't have the entire ontology at its disposal?

LLMs as Teachers

LLMs possess domain knowledge (we hope so!).

Membership queries pose just a few issues , but equivalence queries are problematic.

Asking if the hypothesis ontology is equivalent to the target ontology is not feasible:

- LLMs do not have the target ontology directly.
- Handling long inputs is challenging for LLMs.

ExactLearner

By Konev (2018), proposed for learning EL terminologies.

The learning process assumed that the learner knows the vocabulary (concepts) of the target ontology.

Queries:

Membership - $C \sqsubseteq D$ is Entailed in O (C, D are concepts of O)

The answer will be a positive example otherwise a negative example respecting the target Ontology O.

Equivalence - Ontology H is equivalent to the target O

If equivalent answer “yes” otherwise:

a positive ex. $C \sqsubseteq D$ where $H \not\models C \sqsubseteq D$ or

a negative ex. $E \sqsubseteq F$ where $H \models E \sqsubseteq F$.

ExactLearner

A class of ontologies L is exactly learnable if an algorithm exists that: for any O ∈ L Halts and computes a hypothesis H ∈ L using membership and equivalence queries.

If operating in polynomial time, the class is exactly learnable in polynomial time.

Polynomial Time: Bounded by polynomial p(|O|, C ⊑ D |).

Where C ⊑ D is the largest counter-example found.

Probably Approximately Correct (PAC) for Eq. queries

Objective: Find a hypothesis approximating the target concept using labeled examples from instance space X.

Hypothesis Class H: Set of possible hypotheses considered by the learning algorithm.

Candidate Hypotheses: Each h ∈ H is a potential best approximation based on training data.

Hyper-params:

- ϵ: Maximum allowable error of the hypothesis.
- δ: Probability that the hypothesis error is less than ϵ.

Minimum samples needed m ≥ $\log\left(\frac{|H|}{\gamma}\right) / \epsilon$

Probably Approximately Correct (PAC) for Eq. queries

Estimating Instance Space X :

Consider all combinations of statements:

A∩B⊑C

B⊑∃r.A

∃r.A⊑B

Hypothesis Class Size:
Upper bound on H: $|\mathrm{H}| = |X|^{|A|}$ where |X| is the instance space size and |A| is the number of axioms.

An equivalence query can then be simulated by probing random samples from X.

If a statement is not entailed by the hyp. ontology and it is labeled as `true' by the LLM, then there is no equivalence and the sample is used as a counterexample.
If none counterexample found within *m* samples, the equivalence query is `true'.

From Man. Syntax to NL

Manchester OWL Syntax: A formal language for describing ontologies, close to natural language.

LLM Training: LLMs likely trained more on natural language data, performing better with natural language queries.

```
function translateOWLtoNaturalLanguage(query):
    // Check if the query is of the form "[A] SubClassOf [B]"
    if query matches "[A] SubClassOf [B]":
        A, B = extractConcepts(query, "SubClassOf")
        return "Can " + translateConcept(A) + " be considered a subcategory of " + translateConcept(B)
       + "? Answer only with yes or no."

    // Function to recursively translate concept expressions
    function translateConcept(concept):
        // If the concept is of the form "[A] and [B]"
        if concept matches "[A] and [B]":
            A, B = extractConcepts(concept, "and")
            return translateConcept(A) + " that is also " + translateConcept(B)

        // If the concept is of the form "[r] some [A]"
        else if concept matches "[r] some [A]":
            r, A = extractConcepts(concept, "some")
            return "something that " + r + " " + translateConcept(A)

        // If the concept is a simple concept name, return it as is
        else:
            return concept

    // Helper function to extract concepts based on a delimiter
    function extractConcepts(expression, delimiter):
        // Split the expression by the delimiter to get the components
        parts = split(expression, delimiter)
        return parts[0].trim(), parts[1].trim()
```

Probing LLM with ontology axiom

Standardized Queries: Using Manchester OWL syntax to improve transparency and reproducibility.

Unexpected Responses: LLMs may give arbitrary answers. To mitigate, instruct LLMs to answer with 'true' or 'false'.

Handling Responses: Even with instructions, responses may vary:

- More text than just 'true' or 'false'.
- Both 'true' and 'false' appear.
- Neither 'true' nor 'false' appear.

Solution: Ambiguous responses are classified as 'unknown'.

Correctness and Logical Consistency

Correctness Issues: LLM responses may not reflect the 'truth' or be logically consistent.

Types of Errors:

- $C \sqsubseteq D$ should be 'false', LLM answers 'true'.
- $C \sqsubseteq D$ should be 'true', LLM answers 'false'.
- **Logical inconsistency**: all concepts inclusion in$\{ C1 \sqsubseteq D1 \ldots. Cn \sqsubseteq Dn\}$ are ‘true’ but LLM to $C \sqsubseteq D$ answers 'false'.

Solution: Consider the closure under logical consequence for handling inconsistencies.