Inductive Learning of Logical Theories with LLMs: An Expressivity-graded Analysis

João Pedro Gandarela de Souza[1], **Danilo Carvalho**[2], **André Freitas**[1,2,3]
[1]Idiap Research Institute
[2]National Biomarker Centre, CRUK-MI, University of Manchester
[3]Department of Computer Science, University of Manchester
{firstname.lastname}@idiap.ch, {firstname.lastname}@manchester.ac.uk

Abstract

This work presents a novel systematic methodology to analyse the capabilities and limitations of Large Language Models (LLMs) with feedback from a formal inference engine, on logic theory induction. The analysis is complexity-graded w.r.t. rule dependency structure, allowing quantification of specific inference challenges on LLM performance. Integrating LLMs with formal methods is a promising frontier in the Natural Language Processing field, as an important avenue for improving model inference control and explainability. In particular, inductive learning over complex sets of facts and rules, poses unique challenges for current autoregressive models, as they lack explicit symbolic grounding. While they can be complemented by formal systems, the properties delivered by LLMs regarding inductive learning, are not well understood and quantified. Empirical results indicate that the largest LLMs can achieve competitive results against a SOTA Inductive Logic Programming (ILP) system baseline, but also that tracking long predicate relationship chains is a more difficult obstacle than theory complexity for LLMs.

Introduction

The integration of Large Language Models (LLMs) with formal methods stands out as a promising frontier in the field of Natural Language Processing. It is an avenue for improving model inference control and explainability, by both complementing the content flexibility of Large Language Models (LLMs) with the systematicity of symbolic/formal systems (Quan et al. 2024a,b) and by using well-defined formal settings to assess the underlying inference properties of the model.

Inductive Logic Programming (ILP) is a subfield of symbolic AI which focuses on methods that can derive (generalise) theories to explain observed facts and rules (Muggleton 1991; Nienhuys-Cheng and de Wolf 1997). Addressing inductive learning over complex sets of facts and rules, poses unique challenges for current autoregressive LLMs, as they do not operate over data symbolically, rather combining an extensive set of structural an semantic signals to approximate the most probable answer in a generative fashion.

While such means of problem solving might lack explicit symbolic grounding, LLMs can leverage its large-scale internal representation to support inductive-style inference. Still, the properties delivered by LLMs w.r.t. inductive learning, in particular regarding logic rules and theory induction, are not well understood and quantified.

This paper presents a systematic methodology to evaluate the inductive learning properties (in the context of logic theory induction) of LLMs. It is aimed at answering the following research questions (RQs):

RQ1. To what extent the combination of an LLM with feedback from a formal inference engine can compare to a SOTA inductive logic programming (ILP) system in logic theory induction, w.r.t. inference quality at different expressivity levels?

RQ2. How does the expressivity of the target theories affect inference quality of LLMs for inductive reasoning?

In order to address these RQs, we propose a method for combining iterative theory refinement on stock LLMs (i.e., zero-shot inference (Brown et al. 2020; Radford et al. 2019)), a formal ILP inference engine and a synthetic generator for inductive reasoning datasets, in order to perform systematic evaluations of the LLM induced theories, using state-of-the-art ILP solvers as a baseline. Moreover, to quantify the extent in which LLMs can address ILP tasks, the evaluation is graded w.r.t. the expressivity of the target rulesets. Figure 1 schematises the proposed approach.

This work's main contributions are as follows:

1. (Methodological) A novel method for systematic evaluation of LLM induced logic theories with feedback from a formal inference engine.
2. (Empirical) A detailed empirical analysis on the strengths and limitations of SOTA LLMs regarding logic theory induction generation, according to target ruleset expressivity.
3. (Resources) A reusable and extensible framework for extending and assessing the inductive capabilities of LLMs.

The remainder of the paper is organised as follows: Section "*Inductive Learning, Expressive Power & Datasets*" formalises the relevant ILP concepts, dataset generation, and LLM used in the paper. Section "*Proposed Approach*" describes the proposed method and algorithms. Section "*Experiments*" presents the experimental procedures, results and discussion, followed by related work and conclusions.

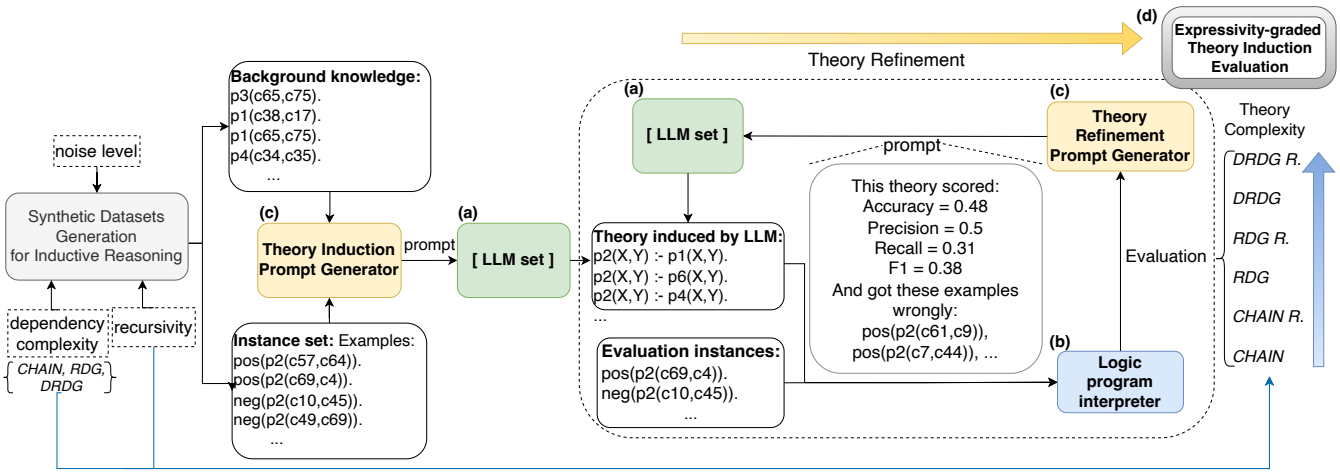

Figure 1: The proposed method to evaluate theory induction with an LLM in Prolog based on background knowledge and training examples. The process starts with a prompt generator (c) that formulates prompts for an LLM (a). Both the background knowledge and training sets are parameterised by different noise and rule expressive power levels: Chain, Rooted Directed Graph (DG), Disjunctive Rooted DG, and Mixed. The LLM generates theories, which are then evaluated by a logic program interpreter (b). The evaluation feedback, including accuracy, precision, recall, and F1 scores, as well as wrongly classified examples, is used to refine the prompts iteratively. We analyse and categorise the generated theories according to their expressive power (d).

Inductive Learning, Expressive Power & Datasets

In this section we introduce the target task and the typology of inductive learning expressivity levels, as well as the dataset generation process.

Inductive Logic Programming

Inductive Logic Programming main objective is to generate logical hypotheses or theories from the available background knowledge, such as facts, rules, and positive and negative examples. Unlike traditional machine learning methods, ILP works directly with logical predicates and rules, using a formal representation that is usually represented using first-order logic (FOL). For example, the fact "parent(john, tom)." means that John is the parent of Tom, and "parent(john, anne)." means that John is the parent of Anne. From that, we can create a rule (*theory*) "sibling(X, Y) :- parent(P, X), parent(P, Y), X $\neq$ Y." . This rule states that if there exists a parent who has both X and Y as children, and X and Y are not identical, then X and Y are considered siblings. Deriving rules from a set of examples is a process known as *theory induction*. This task can be formally defined as follows:

Given:

- *Background Knowledge (BK):* A set of logical clauses that represent prior knowledge about the domain.
- *Positive Examples (E^+):* A set of ground facts (instances) which the learned theory should entail (i.e., these are examples that should be true according to the theory).
- *Negative Examples (E^-):* A set of ground facts which the learned theory should not entail (i.e., these are examples that should be false according to the theory).

Goal: Find a hypothesis H (a set of logical clauses) such that:

1. **Completeness:** For every example $e \in E^+$:
$$H \cup BK \models e$$
Meaning the hypothesis H, together with the background knowledge BK, should entail all positive examples.
2. **Consistency:** For every example $e \in E^-$:
$$H \cup BK \not\models e$$
Meaning the hypothesis H, together with the background knowledge BK, should not entail any negative examples.

Formally: An ILP system seeks a hypothesis H that satisfies:

$$\forall e \in E^+,\ BK \cup H \models e$$
$$\forall e \in E^-,\ BK \cup H \not\models e$$

Conclusion: The learned hypothesis H should thus be a logical theory that explains the positive examples and excludes the negative examples, based on the given background knowledge.

Theory Expressivity

Inductive learning can be organised according to different classes of expressive power, which involves a typology of the structural complexity of the problem. Moreover, variables such as the amount and type of noise within the evidence set (such as the number of incorrect or missing facts or completeness levels) can be integrated within this setting. Following the typology of (Cornelio and Thost 2021)

four base categories of rules can be introduced based on their dependencies: Chain, Rooted Directed Graph (RDG), Disjunctive Rooted DG, and Mixed. Each category represents a hierarchical generalisation, with each step encompassing a broader range of structures. Starting with CHAIN, which represents a linear composition of predicates, RDG is a generalisation of CHAIN, meaning it includes all chain structures but also accommodates more complex dependencies. Moving up, DRDG generalises RDG by incorporating directed relationships, thus offering a more extensive representation of dependencies. Finally, Mixed is a DRDG with recursion, containing connected components from CHAIN, RDG, and DRDG. Each progression from CHAIN to MIXED represents a step towards greater inclusivity and complexity in the types of structures captured.

The expressive power of each class and their characteristics are summarised in Table 1. A detailed description of each class is provided in the supplementary material[1].

Category	# Parents	Recursive	Alt. rules
CHAIN	1	No	No
CHAIN REC.	1	Yes	No
RDG	1 – *	No	No
RDG REC.	1 – *	Yes	No
DRDG	1 – *	No	Yes
DRDG REC.	1 – *	Yes	Yes
MIXED	1 – *	Yes	Yes

Table 1: Characteristics for each dataset category. *#Parents* refers to the number of times a predicate appears as the head of a rule in the context of rule learning and logical induction. It indicates the number of rules through which each rule can deduce relevant facts. *Recursive* refers to whether a predicate in the head of a rule can also occur in the body of the same rule. *Alt. rules* indicate whether a predicate can be deduced by alternative rules.

Dataset Synthesis

In order to generate datasets for rigorous analysis, this study employed the RuDaS tool (Cornelio and Thost 2021) to systematically vary parameters such as noise, open-world degree, and missing data. By adjusting these factors in conjunction with the *category* parameter, it is possible to ensure comprehensive coverage of different structural configurations and expressivity levels.

For each configuration, the following settings were applied:

- The minimum and maximum number of Directed Acyclic Graphs (DAGs), which refer to the range of distinct rule structures generated within each dataset, were both set to 1 (`mindags` = 1, `maxdags` = 1).
- Noise levels, which include the addition of irrelevant facts or the removal of relevant ones that do not support the rules, were systematically varied at intervals of 0.1, 0.2, and 0.3.
- The percentage of missing data (`missing`), which determines the proportion of data entries - specifically consequences - that are intentionally left out, and open world degree (`owa`) were similarly varied across 0.1, 0.2, and 0.3.
- The category parameter was set to cover all classes of expressivity described in the previous section, and listed in Table 1. The distribution of each category in the synthesised dataset is an independent hyperparameter, discussed in the experiments section.

Further details regarding the dataset generation can be found in the supplementary material[1].

[1]A version of this paper with the supplementary material can be found at https://arxiv.org/pdf/2408.16779.

Proposed Approach

The proposed approach can be divided in two parts: *iterative refinement* and *graded evaluation*. They form a systematic evaluation loop, covering all the expressivity classes described in the previous section, for a given set of LLMs and dataset synthesis parameters.

Iterative Refinement

Consists of an iterative refinement loop that alternates between the generation of a theory by a language model and the evaluation of said theory through a formal interpreter. It is comprised of the following components, as illustrated in Figure 1:
(a) A language model capable of generating a theory H, based on background knowledge, positive and negative examples, and hypothesis search, provided as a prompt text in a logic program language. Typically an LLM with structured (e.g., code) generation capabilities. **(b)** A logic program interpreter. We use Prolog (Warren et al. 2023) as the logic program language. **(c)** A prompt generation component, that interleaves logical programming language with natural language queries designed to drive the theory induction responses. The logical programming language expresses background knowledge and the relevant outputs of the program interpreter. **(d)** An evaluation module, that uses the logic program interpreter to execute the generated theory H as logical rules, and computes a set of evaluation metrics. Given: *background knowledge (BK)*, *positive examples* (E^+), *negative examples* (E^-), and assuming a language model $\mathcal{LM}$ which can find a hypothesis H (a set of logical clauses) such that it satisfies the conditions of completeness (for every example $e \in E^+$, $H \cup BK \models e$) and consistency (For every example $e \in E^-$, $H \cup BK \not\models e$).

1. **Context Representation**: Represent the input to the language model as a combination of background knowledge and examples: Context $=$ encode(BK, E^+, E^-).

2. **Theory Generation**: From a background knowledge set of clauses sampled from a knowledge base dataset, including positive and negative examples, a prompt is created for the LM to induce a theory as Prolog code, i.e. using the language model to generate a set of logical clauses (hypothesis H): $H = \mathcal{LM}$(Theory Prompt + Context).

3. **Evaluation of Hypothesis**: Checking for the completeness and consistency conditions:
True Positives (TP): The number of positive examples correctly entailed by the hypothesis.
$TP = |\{e \in E^+ \mid BK \cup H \models e\}|$

False Positives (FP): The number of negative examples incorrectly entailed by the hypothesis.
$FP = |\{e \in E^- \mid BK \cup H \models e\}|$

False Negatives (FN): The number of positive examples not entailed by the hypothesis.
$FN = |\{e \in E^+ \mid BK \cup H \not\models e\}|$

True Negatives (TN): The number of negative examples correctly not entailed by the hypothesis.
$TN = |\{e \in E^- \mid BK \cup H \not\models e\}|$

from which accuracy (ACC) ($\frac{TP+TN}{TP+TN+FP+FN}$), precision (P) ($\frac{TP}{TP+FP}$), recall (R) (Recall $= \frac{TP}{TP+FN}$) and F1-score (F1) ($2 \cdot \frac{\text{Precision} \cdot \text{Recall}}{\text{Precision}+\text{Recall}}$) can be generated.

4. **Theory refinement**: Following an initial evaluation, the LM is tasked to refine the induced theory iteratively. Each refinement round involves adjusting the theory based on feedback from the Prolog interpreter validation. The refinement aims to improve the theory's performance by addressing misclassifications and enhancing its predictive capabilities. If H does not satisfy completeness and consistency, update the input representation based on feedback and generate a new hypothesis using the language model, given the *Feedback Context* $\leftarrow \{FP, FN, P, R, F1, ACC\}$ and the final prompt input *Input* $\leftarrow$ (Refinement Prompt + Context + Feedback Context): $H \leftarrow \mathcal{LM}(\text{Input})$.

The main loop of the algorithm continues until the evaluation metrics meet the defined thresholds or the maximum number of iterations is reached. In each iteration, the language model generates a theory based on the current prompt. This generated theory is then evaluated using the logic program interpreter, in our case Prolog, resulting in a validation set. Evaluation metrics are computed from these validation results and stored. Based on the feedback from the validation, a new prompt is generated by incorporating the initial knowledge base sample, the current theory, and the validation feedback. Our approach removes recursive rules from the LLM-induced theory before evaluation. The refinement loop is summarised in Algorithm 1. The process starts by sampling facts from the knowledge base dataset to create kb. An initial prompt is then generated using these sampled facts, denoted as $prompt$.
5. **Termination**: The process continues iteratively until a maximum number of iterations is reached.

Algorithm 1: Iterative LM theory refinement

Define: KB as the background knowledge dataset.
Define: $PGen$ as the prompt generator.
Define: Exs as the positive and negative examples.
Define: LM as the language model.
Define: PL as the logic program interpreter.
Define: $Eval$ as the evaluation module.
Define: M as the evaluation metrics set.
Define: Max_{iter} as the maximum number of iterations.
Define: $MT_{tresh} \leftarrow Map : M \rightarrow \mathbb{R}$ as the evaluation metrics treshold.

Let $prompt \leftarrow PGen(KB, Examples)$
Let $iter \leftarrow 0$
Let $results \leftarrow Map : M \rightarrow \mathbb{R}$
while $(\exists m \in results : results[m] < MT_{tresh}[m]) \wedge (iter < Max_{iter})$ **do**
  $theory \leftarrow LM(prompt)$
  $results \leftarrow Eval(Examples)$
  $prompt \leftarrow PGen(KB, theory, Exs)$
  $iter \leftarrow iter + 1$
end while

Graded Evaluation

A synthetic data generator is used to control for the input parameters of the ruleset expressive power, namely: categorical distribution (CHAIN, RDG, DRDG, etc.), background knowledge, positive examples and negative examples as well as the amount of noise introduced within the dataset.

1. **Categorised Learning Sets**: Consisting of C: set of ruleset expressivity classes (e.g., CHAIN, RDG, DRDG, etc.), N: set of noise levels, S: number of samples per combination of C and N. For each $c \in C$ and $n \in N$, generate S datasets $\{D_{c,n,i} \mid i = 1, \ldots, S\}$ where each dataset $D_{c,n,i}$ includes:

$$D_{c,n,i} = (\text{BK}_{c,n,i}, E^+_{c,n,i}, E^-_{c,n,i}, \text{noise}_{c,n})$$

2. **Hypothesis Generation and Evaluation**: For each dataset $D_{c,n,i}$, use a learning algorithm to generate a hypothesis $H_{c,n,i}$, tracking the F1 score $F_{c,n,i}$ and processing time $T_{c,n,i}$ at each iteration and recording the best F1 score $\text{F1}_{c,n,i}$ and corresponding processing time $\text{Time}_{c,n,i}$:

$$\text{F1}_{c,n,i} = \max(F_{c,n,i})$$

$$\text{Time}_{c,n,i} = \text{time until } \max(F_{c,n,i})$$

3. **Aggregation**: The information is then aggregated by complexity category and noise level for all the samples, averaging times and F1 scores, to obtain the complete graded evaluation statistics. For each combination of $c \in C$ and $n \in N$, compute the average F1 score and average processing time:

$$\overline{\text{F1}}_{c,n} = \frac{1}{S} \sum_{i=1}^{S} \text{F1}_{c,n,i}$$

$$\overline{\text{Time}}_{c,n} = \frac{1}{S} \sum_{i=1}^{S} \text{Time}_{c,n,i}$$

Experiments

In order to answer the research questions, a set of experiments was elaborated to systematically analyse the the-

ory inducing capabilities of a set of most popular open-source LLMs and two versions of GPT, with the proposed approach, having the state-of-the-art ILP system *Popper* (Cropper and Morel 2021) as a baseline. The tests covered all data categories discussed on the section "*Inductive Learning, Expressive Power & Datasets*", allowing a graded analysis w.r.t. the expected level of induction expressivity and tolerated noise.

Experimental Setup & Dataset

For each data category, five datasets were generated using *RuDaS* (Cornelio and Thost 2021). The size of each dataset was set to XS ($min = 50$, $max = 100$, $support = 3$), and noise, missing, and open world were all set to 0.1, then all set to 0.2, and finally all set to 0.3. This resulted in 105 datasets in total, with 35 datasets for each rate. Subsequently, two methods are used to induce a theory for each dataset: (1) employing *Popper*, with NuWLS (Chu, Cai, and Luo 2023) and WMaxCDCL, varying its time limit parameter from 10 to 800 seconds; (2) applying the proposed iterative LM theory refinement method (Section "*Proposed Approach*"), with parameters $Max_{iter} = 4$ and $MT_{thresh} = 1.0$. Three different LLM models were used for (2): Open AI[2]'s model *GPT-4o*[3], Mistral AI[4]'s *Mixtral-8x7B*[5] (Jiang et al. 2023), and Google's *Gemma*[6] (Team et al. 2023).

Table 2 presents a comprehensive overview of statistical metrics pertaining to each category of data, in order of complexity (except *MIXED*).

Categories	Facts	Positive	Negative
CHAIN	67.6 ± 5.4	23.6 ± 6.5	4.8 ± 2.1
CHAIN REC.	54.2 ± 14.1	19.6 ± 7.8	4.4 ± 2.5
RDG	60.2 ± 9.8	18.6 ± 12.3	4.2 ± 3.4
RDG REC.	63.2 ± 9.0	16.6 ± 4.7	3.4 ± 1.3
DRDG	61.2 ± 14.1	31.0 ± 26.0	8.0 ± 8.2
DRDG REC.	54.2 ± 12.5	34.0 ± 22.2	9.0 ± 6.2
MIXED	54.4 ± 18.1	24.2 ± 12.3	4.8 ± 2.7

Table 2: Statistics for each dataset category. A detailed description of each can be found in Section "*Inductive Learning, Expressive Power & Datasets*". The mean $\pm$ standard deviation (mean $\pm$ std) for each category is provided

We computed the average F1-score for each category, taking into account the level of noise, open world scenarios, and missing facts. The mean values reported are based on the results obtained from the theory that was generated from the train set and evaluated on the test set.

The experiment used the OpenAI service for GPT models. For Popper, Llama3-8B-Instruct, Gemma-7B-It and Mixtral-8x7B-Instruct-v0.1, it was conducted on a computer with an Intel(R) Xeon(R) Gold 5217 CPU @ 3.00GHz, 188GB RAM, and 2x NVIDIA RTX A6000 (48GB VRAM) GPUs. The software used was CUDA 12.3, PyTorch 2.2.2, and Transformers 4.41.2. Prompt templates used were included in the supplementary material[1].

[2]https://openai.com/

[3]https://openai.com/index/hello-gpt-4o

[4]https://mistral.ai/

[5]https://huggingface.co/mistralai/Mixtral-8x7B-Instruct-v0.1

[6]https://huggingface.co/google/gemma-7b-it

Results & Discussion

Overall results for F1 are presented in Figure 2. We additionally report on processing times as a measure of practical interest in Figure 3. We present the values obtained in detail in the supplementary material[1] (Tables 4 and 5) *Gemma-7B-It* results are not included as it failed to generate valid theories. The results reveal significant insights into LLM capabilities and limitations w.r.t. theory induction, which are summarised as follows:

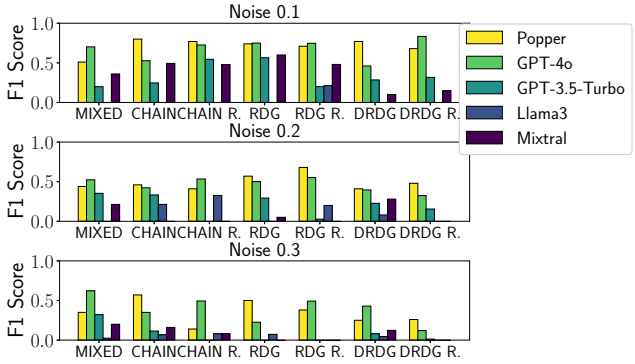

Figure 2: F1 score trends across categories. Different models (GPT-4o, Llama3 8b instruct, Popper, and Mixtral-8x7B-Instruct-v0.1) under varying noise levels and categories reveal distinct performance patterns. GPT-4o demonstrates stable accuracy yet sensitivity to noise, particularly in complex rule-based categories like RDG and DRDG. Mixtral-8x7B-Instruct-v0.1 exhibits mixed results with notable variability across categories particularly in more complex tasks. Llama3 8b instruct delivers lower scores, indicating challenges in reasoning and theory generation.

LLMs can achieve competitive performance against the baseline, specially at higher noise levels. The larger scale models (GPT3.5, 4) demonstrate more resilience to noise and consistent F1 across the different categories, as indicated in Figure 2, with an average F1-score difference of ± 0.25 against Popper. This answers the overall quality part of **RQ1**.

Inducing theories on long relation chains is the major obstacle for LLMs, rather than the expressivity level. With the CHAIN category being the least expressive and one of the most consistently solved by Popper, but none of the tested LLMs was able to overcome the baseline performance at all noise levels (Figure 2 *CHAIN* category). This suggests that such models have a limited capacity of tracking long relationship chains of independent predicates. This addresses a part of the answer for **RQ2**.

Increasing iteration limits does not monotonically improve results for LLMs. Upon increasing the iteration

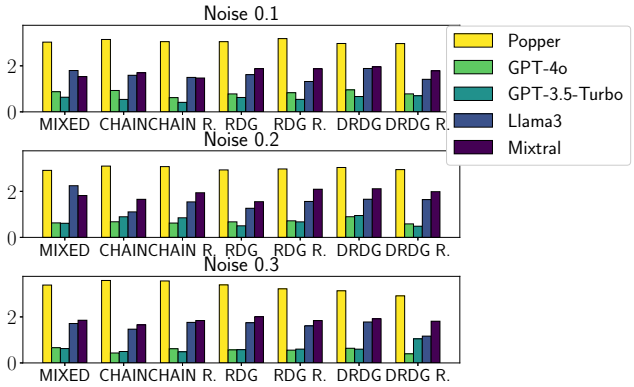

Figure 3: Performance on time consumption trends across categories using a logarithmic scale ($log(time\ (s))$). The data consistently shows that LLM outperforms Popper in all intervals. The results however do not represent a measure of efficiency, as the computational resources employed are vastly different across methods.

limits from 1 to 4, it was found that the metrics can either increase or decrease non-monotonically. No significant overall performance improvements were observed beyond 3 iterations. Thus Max_{iter} was set to 4 and the iteration with the best accuracy is taken as the final result.

Performance is remarkably lower on more expressive rule sets at moderate noise levels. Responses for more expressive categories, such as RDG and DRDG display higher variance and LLM hallucination artifacts, such as valid rules containing predicates that do not exist in the rule set. We present an error analysis in the section "*Error analysis*". For instance, a comparison of the results for the RDG category generated by GPT-4o under noise levels set to 0.1 and 0.3 reveals a significant decline in performance, with the F1-score dropping from 0.75 to 0.22. A comparable pattern is observed with GPT-3.5 Turbo for RDG and DRDG and Mixtral RDG in the presence of elevated noise levels, with GPT-3.5 Turbo scores going from 0.56 to 0.0, 0.28 to 0.08, and Mixtral-8x7B going from 0.60 to 0.0. This complements the answer to **RQ2**. Further details are included in the supplementary material[1].

Induction capability varies substantially across models. Using the same inputs resulted in vastly different responses from different models, suggesting critical influence from model size in parameters. Figure 2 illustrates this: when comparing GPT-4o, Mixtral-8x7B and Llama3 at noise levels set to 0.1 and 0.3 respectively, the consistency in generating a valid theory correlates to their relative size.

At a noise level of 0.1, GPT-4o's F1 score is almost twice that of the GPT-3.5-Turbo in average, and at a noise level of 0.3, the difference increases to a ratio of 4, indicating substantially higher noise resiliency. The performance gap is more pronounced when comparing with Llama3-8B, where GPT-4o F1 score is 21 times higher at the lowest noise setting.

Mixtral-8x7B-It-v0.1 performs similarly to GPT-3.5-Turbo at lower noise levels, scoring 13.4% higher in average at 0.1 noise. However, its performance becomes less stable at higher noise levels. It consistently outperforms Llama3-8B-it, at 0.1 noise, with a F1-score 11 times higher in average.

Model size does not correlate to noise resilience Despite being able to achieve higher scores than GPT-3.5 and Mixtral-8x7B in some of the tests (e.g., RDG-R $@noise = 0.1$, CHAIN-R $@noise = 0.2$) and scoring higher on intermediate noise than on low noise, Llama3-8B did not consistently generate valid theories. On the other hand, Mixtral-8x7B, a much larger model, is particularly susceptible to noise, with an average F1-score drop of over 0.8 from $noise = 0.1$ to $noise = 0.2$ and a monotonic performance reduction with the increase of the noise level.

Regarding the parameterisation of each method, some higher-level observations on the trade-off between time and inference quality can be summarised as follows:

Computational scalability allow LLMs to operate at substantially lower times. While Popper consists of a combination of serial algorithms, the parallel nature of transformer-based LLMs allows them to operate at times about 3 orders of magnitude lower, given enough computational resources. This can be observed in Figure 3, for all classes and all tested noise intervals.

Error Analysis

The errors found in the evaluation of the generated theories could be separated in two categories: *syntactic* and *logical*. *Syntactic* errors occur when the generated response does not match the logic programming language grammar. For example, the following response:

```
theory :-
    p(X, Y), pos(p0(X, Y)) - positive.
    p(X, Y), neg(p0(X, Y)) - negative.
    \+ p(X, Y), pos(p0(X, Y)) - false.
    \+ p(X, Y), neg(p0(X, Y)) - true.
```

is not valid Prolog and will fail evaluation.

Logical errors occur when the generated response has correct grammar, but cannot be induced from the examples. Consider the following Prolog theory:

```
theory :-
    p(X, Y) :- p1(X, Y); p3(X, Y);
    p4(X, Y); p7(X, Y); p8(X, Y);
    p0(X, Y),
    not neg(p(X, Y)),
    (pos(p(X, Y)) - true; fail).
```

The response contains the head of the clause "theory," as well as the predicates "p" and "pos", which do not exist in the BK. Table 3 presents a distribution of error categories for the analysed models. A more detailed analysis of the models outputs is included in the supplementary material[1].

Model	# Syntactic	Logical
GPT-4o	0%	100%
GPT3.5	0%	100%
Llama3-8B	46%	54%
Mixtral-8x7B	20%	80%
Gemma-7B-it	100%	0%

Table 3: Error distribution for each of the evaluated models. Gemma-7B-it did not produce valid Prolog.

Related Work

Neural symbolic computation combines neural networks with symbolic methods to enhance AI reasoning capabilities. Yang and Cohen (2017) introduced Neural Logic Programming, An end-to-end differentiable model integrating neural networks with logic programming. Within the LLM-Symbolic space, Wan et al. (2024) developed LogicAsker, which evaluates and improves LLMs' logical reasoning using propositional and predicate logic. It identifies reasoning failures and enhances capabilities through in-context learning. Within the context of symbolic toolformers over LLMs, Quan et al. (2024a; 2024b) proposed methods of improving explanatory reasoning with the support of formal iterative cycles using both logical solvers and theorem provers for supporting more controlled step-wise reasoning.

Despite these advancements at the interface of LLM-based reasoning and formal controls, it is unclear the extent and the conditions in which LLMs can perform formal reasoning (Huang and Chang 2023). Sinha et al. (2019) introduced CLUTRR, a benchmark assessing LLMs' structural learning by inferring kinship relations in stories, requiring relationship extraction and logical rule inference. Zhu et al. (2024) proposed the Hypotheses-to-Theories (HtT) framework to improve LLM reasoning by learning explicit rules in two stages: generating and verifying rules (induction) and using the obtained rule library for reasoning (deduction). HtT enhances relational and numerical reasoning and concept learning.

Madaan et al. (2023) introduces a novel technique for improving machine learning models through iterative refinement. This approach allows models to improve their performance by continuously evaluating and adjusting their predictions based on self-generated feedback. By critiquing their own outputs, models can identify errors and make corrections over successive iterations, leading to increased accuracy and robustness across different tasks. Our work builds upon this approach by employing an ILP solver to evaluate and then using its output as critique for the refinement process.

In a related study (Dziri et al. 2023), the authors investigate the limitations of transformer models in handling composition tasks. Their results show that, despite their strengths, transformers face significant challenges in dealing with compositionality, which involves understanding and generating complex structures from simpler components. This limitation highlights the need for innovative approaches, such as self-refining, to further enhance the capabilities of machine learning models.

In contrast, our work focuses on the still under-explored area of assessing and controlling inductive learning/inference capabilities of LLMs. These contributions integrate LLMs and formal logic for robust theory induction and allows a graded analysis of LLM capabilities, with respect to theory induction complexity.

Conclusion

In this study we thoroughly investigate the integration of state-of-the-art formal theory induction within the context of large language models (LLMs), aiming to elucidate the extent in which LLMs can systematically perform inductive learning for theories spanning across different expressivity levels. At the heart of this exploration lies the recognition of relational data's inherent semantic depth, stemming from its symbolic representations. The empirical results presented here have indicated the ability of LLMs to address inductive learning tasks, with the largest LLMs achieving competitive results against the algorithmic SOTA with better tolerance to higher noise levels, which can be attributed to their semantic flexibility. This flexibility however has certain limitations, as we found that tested language models are more limited by their capacity of tracking long relationship chains of independent predicates than by the dependency complexity of the rule sets (disjunctiveness, recursivity).

As future work we plan to use larger datasets to test the scalability of the proposed approach, allowing the assessment of the performance of LLMs across a broader range of domain-specific and agnostic scenarios. Additionally, we will investigate the impact of LLMs for bootstrapping the initial generation of theories.

Limitations

While the proposed evaluation methodology aims to cover a wide range of logic theory induction expressivity levels, it is limited in its resolution to the categories specified by (Cornelio and Thost 2021), and does not quantify other ruleset characteristics, such as *workspace size* or *unification rate* in the case of Prolog (Dikovsky 1993).

The methodology compares all models under the same inputs. Therefore it is not concerned with extracting maximum performance of any given model, but obtaining a relative assessment of their fundamental capabilities. This means that the empirical analysis reported scores should not be taken as a measure of SOTA performance.

Furthermore, the time gains demonstrated in the experiments are presented as an achievable result, conditioned to the combination of software and hardware indicated in the paper and the services provided by third-parties (e.g., OpenAI). They should not be interpreted as a measure of computational efficiency.

Acknowledgements

This work was partially funded by the Swiss National Science Foundation (SNSF) project NeuMath (200021_204617)[7], by the CRUK National Biomarker Centre, and supported by the Manchester Experimental Cancer

[7]https://data.snf.ch/grants/grant/204617

Medicine Centre and the NIHR Manchester Biomedical Research Centre.

References

Brown, T. B.; Mann, B.; Ryder, N.; Subbiah, M.; Kaplan, J.; Dhariwal, P.; Neelakantan, A.; Shyam, P.; Sastry, G.; Askell, A.; Agarwal, S.; Herbert-Voss, A.; Krueger, G.; Henighan, T.; Child, R.; Ramesh, A.; Ziegler, D. M.; Wu, J.; Winter, C.; Hesse, C.; Chen, M.; Sigler, E.; Litwin, M.; Gray, S.; Chess, B.; Clark, J.; Berner, C.; McCandlish, S.; Radford, A.; Sutskever, I.; and Amodei, D. 2020. Language models are few-shot learners. In *Proceedings of the 34th International Conference on Neural Information Processing Systems*, NIPS '20. Red Hook, NY, USA: Curran Associates Inc. ISBN 9781713829546.

Chu, Y.; Cai, S.; and Luo, C. 2023. NuWLS: Improving Local Search for (Weighted) Partial MaxSAT by New Weighting Techniques. *Proceedings of the AAAI Conference on Artificial Intelligence*, 37(4): 3915–3923.

Cornelio, C.; and Thost, V. 2021. Synthetic Datasets and Evaluation Tools for Inductive Neural Reasoning. In *Proceedings of the 30th International Conference on Inductive Logic Programming, ILP2020-21 @ IJCLR*.

Cropper, A.; and Morel, R. 2021. Learning programs by learning from failures. *Machine Learning*, 110(4): 801–856.

Dikovsky, A. J. 1993. On computational complexity of Prolog programs. *Theoretical Computer Science*, 119(1): 63–102.

Dziri, N.; Lu, X.; Sclar, M.; Li, X. L.; Jian, L.; Lin, B. Y.; West, P.; Bhagavatula, C.; Bras, R. L.; Hwang, J. D.; et al. 2023. Faith and Fate: Limits of Transformers on Compositionality. *arXiv preprint arXiv:2305.18654*.

Huang, J.; and Chang, K. C.-C. 2023. Towards Reasoning in Large Language Models: A Survey. In Rogers, A.; Boyd-Graber, J.; and Okazaki, N., eds., *Findings of the Association for Computational Linguistics: ACL 2023*, 1049–1065. Toronto, Canada: Association for Computational Linguistics.

Jiang, A. Q.; Sablayrolles, A.; Mensch, A.; Bamford, C.; Chaplot, D. S.; Casas, D. d. l.; Bressand, F.; Lengyel, G.; Lample, G.; Saulnier, L.; et al. 2023. Mistral 7B. *arXiv preprint arXiv:2310.06825*.

Madaan, A.; Tandon, N.; Gupta, P.; Hallinan, S.; Gao, L.; Wiegreffe, S.; Alon, U.; Dziri, N.; Prabhumoye, S.; Yang, Y.; Welleck, S.; Majumder, B. P.; Gupta, S.; Yazdanbakhsh, A.; and Clark, P. 2023. Self-Refine: Iterative Refinement with Self-Feedback. arXiv:2303.17651.

Muggleton, S. 1991. Inductive logic programming. *New generation computing*, 8: 295–318.

Nienhuys-Cheng, S.-H.; and de Wolf, R. 1997. *What is inductive logic programming?* Springer.

Quan, X.; Valentino, M.; Dennis, L. A.; and Freitas, A. 2024a. Enhancing Ethical Explanations of Large Language Models through Iterative Symbolic Refinement. arXiv:2402.00745.

Quan, X.; Valentino, M.; Dennis, L. A.; and Freitas, A. 2024b. Verification and Refinement of Natural Language Explanations through LLM-Symbolic Theorem Proving. arXiv:2405.01379.

Radford, A.; Wu, J.; Child, R.; Luan, D.; Amodei, D.; Sutskever, I.; et al. 2019. Language models are unsupervised multitask learners. *OpenAI blog*, 1(8): 9.

Sinha, K.; Sodhani, S.; Dong, J.; Pineau, J.; and Hamilton, W. L. 2019. CLUTRR: A Diagnostic Benchmark for Inductive Reasoning from Text. In Inui, K.; Jiang, J.; Ng, V.; and Wan, X., 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)*, 4506–4515. Hong Kong, China: Association for Computational Linguistics.

Team, G.; Anil, R.; Borgeaud, S.; Wu, Y.; Alayrac, J.-B.; Yu, J.; Soricut, R.; Schalkwyk, J.; Dai, A. M.; Hauth, A.; et al. 2023. Gemini: a family of highly capable multimodal models. *arXiv preprint arXiv:2312.11805*.

Wan, Y.; Wang, W.; Yang, Y.; Yuan, Y.; Huang, J.; He, P.; Jiao, W.; and Lyu, M. R. 2024. A & B == B & A: Triggering Logical Reasoning Failures in Large Language Models. *CoRR*, abs/2401.00757.

Warren, D. S.; Dahl, V.; Eiter, T.; Hermenegildo, M. V.; Kowalski, R.; and Rossi, F. 2023. *Prolog: The Next 50 Years*, volume 13900. Springer Nature.

Yang, F.; Yang, Z.; and Cohen, W. W. 2017. Differentiable learning of logical rules for knowledge base reasoning. *Advances in neural information processing systems*, 30.

Zhu, Z.; Xue, Y.; Chen, X.; Zhou, D.; Tang, J.; Schuurmans, D.; and Dai, H. 2024. Large Language Models can Learn Rules. arXiv:2310.07064.