Rule Learning in the LLM Era: Foundations, Techniques, and Applications

Xiang Gao
Intuit

Siyuan Wang
USC

Chenyan Xiong
CMU

https://github.com/golsun/rule_learning

Outline

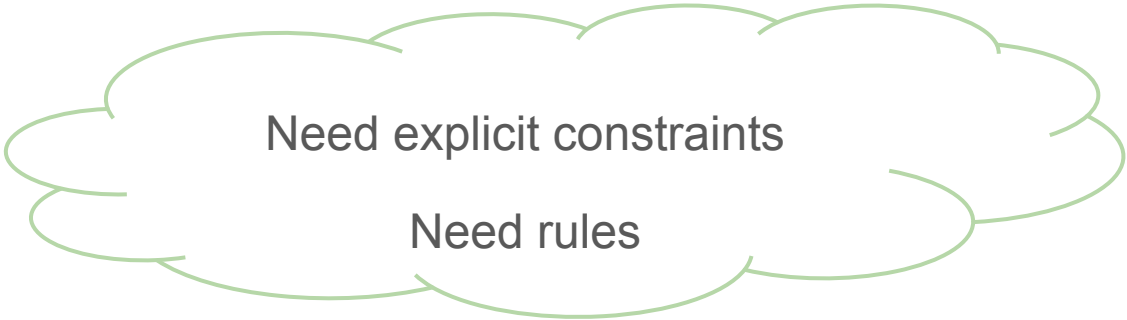
- Introduction
- Foundations
- Rule Discovery
- Rule-Augmented Generation
- Industry Applications
- Open Questions

Why Do LLMs Still Fail in Practice?

- Fluent, confident outputs can violate real rules
- Failures are often silent and hard to detect
- Especially problematic in rule-governed domains:
  - Regulations
  - Policies
  - Domain conventions

High-Stakes Domains Expose LLM Weaknesses

- Many domains have explicit rules:
  - Regulations
  - Safety constraints
  - Procedural guidelines
- LLMs:
  - Do not natively enforce rules
  - Cannot guarantee compliance
  - Often “sound right” while being wrong



Do LLMs already master rules? Probably no

"Through carefully designed intervention experiments on five math tasks, we confirm that transformers are performing case-based reasoning"

Case-Based or Rule-Based: How Do Transformers Do the Math?

Yi Hu [1] Xiaojuan Tang [1 2] Haotong Yang [1] Muhan Zhang [1 2]

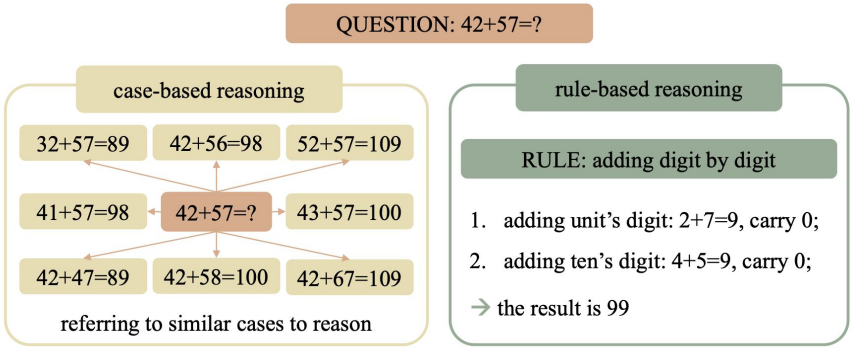


Figure 1. Illustrations of case-based and rule-based reasoning.

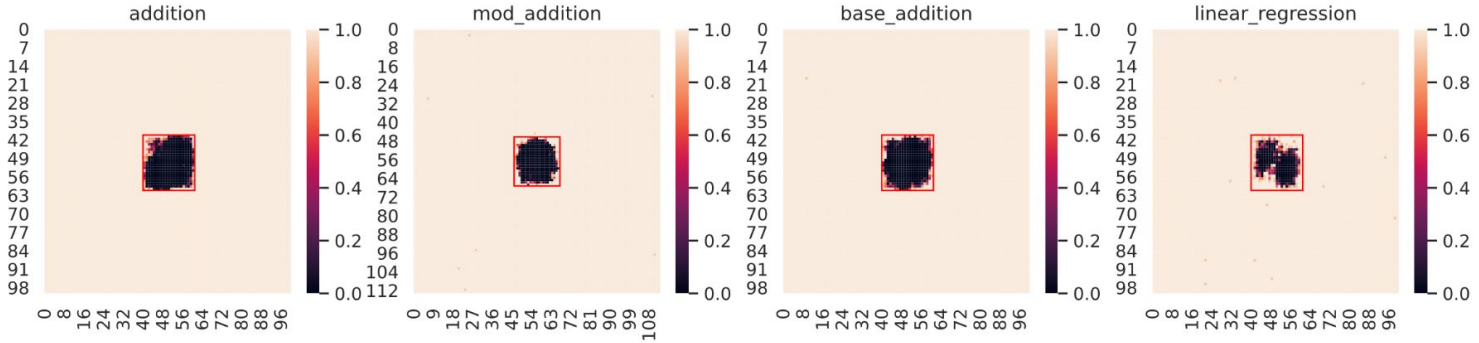


Figure 2. Accuracy of Leave-Square-Out method on addition, modular addition, base addition, and linear regression. The vertical and horizontal axes are a and b, respectively. The area inside red boxes represents the test squares. During generation, we set the model temperature to 1 and sample 10 generations to evaluate the accuracy on each test point. We only leave one test square out in this experiment. The square center (a_k, b_k) is (50, 50) for addition, base addition and linear regression and (56, 56) for modular addition.

Case-Based or Rule-Based: How Do Transformers Do the Math?

Why Bring Rules Back?

- **Pillar 1 Knowledge Grounding**
  - Problem: Black-box Models lack explicit structured knowledge
  - Solution: Discovered rules explicitly inject domain knowledge
  - Impact: Enhanced knowledge retrieval and reasoning capabilities

- **Pillar 2: Interpretability & Control**
  - Problem: Reasoning processes are opaque and untraceable
  - Solution: Rules make decision paths transparent
  - Impact: Debuggable, auditable AI systems

- **Pillar 3: Safety & Compliance**
  - Problem: Behavior misaligned with human preferences and safety standards
  - Solution: Rule-guided generation and verification
  - Impact: Adherence to domain regulations (healthcare, finance, law)

- **Pillar 4: Continual Adaptation**
  - Problem: Models fail to evolve with dynamic and unknown environments
  - Solution: Rules provide protocol for learning from success v.s. failures
  - Impact: Self-improving, adaptive systems

Rules and LLMs: A Two-Way Street

- Rules → LLMs
  - Ground domain knowledge
  - Guide multi-step reasoning
  - Enforce safety & compliance

- LLMs → Rules
  - Discover rules from data
  - Extract preferences & constraints
  - Adapt rules post-deployment

Rule Learning Is Not New — But the Context Is

- Classic foundations:
  - Explanation-based learning
  - Inductive Logic Programming (ILP)
- Why they stalled:
  - Manual knowledge engineering
  - Limited coverage
- What changed:
  - LLMs as flexible pattern learners
  - Natural language as a rule interface

Tutorial Scope & Takeaways

- Introduce foundations of rule learning
- Survey modern techniques for:
  - Rule discovery
  - Rule-augmented generation
- Show real-world applications
- Discuss open challenges & research directions

Foundations

Different Kinds of Rules

- Factual rules (domain knowledge)
- Procedural rules (how to act)
- Preference rules (what is better or worse)
- Safety / compliance rules (what is forbidden)

IF patient has fever AND rash THEN likely high white blood cell count.

IF obstacle detected THEN turn left and move forward.

If two actions are both legal, prefer the one with higher expected profit *unless* risk exceeds a fixed threshold.

No vehicle shall exceed the posted speed limit.

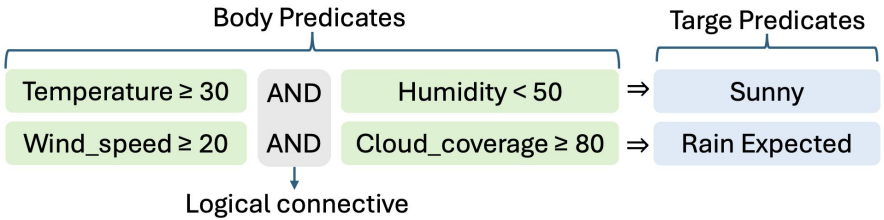
Rule Representations

- Logical rules (e.g., first-order logic)
- Graph-based rules
- Programmatic constraints
- Natural language rules

Rule Template (First Order Logic)	Rule Template (Natural Language)
$\forall x, condition(x) \implies conclusion$	If __, then __.
$\exists x, condition(x) \implies conclusion$	There exists __, which __.
$\forall x, condition(x)\ [\wedge\ condition(x)]^{+} \implies conclusion$	If __ and __, then __.
$\forall x, condition(x)\ [\vee\ condition(x)]^{+} \implies conclusion$	If __ or __, then __.

If entity A →relation1→ B and B →relation2→ C, then infer A →relation3→ C

```
if battery_level < 20%:
    reduce_speed()
    seek_charging_station()
```



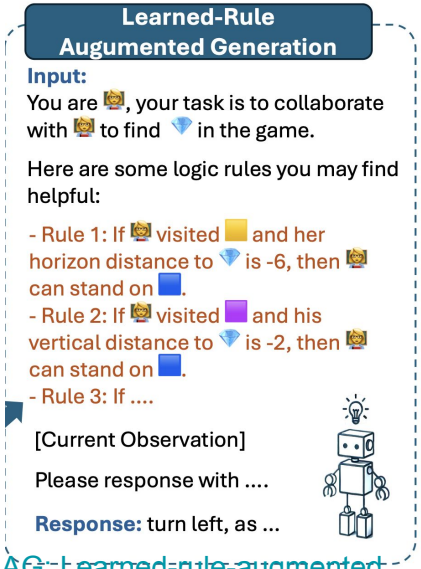
“If a customer has an outstanding balance past 60 days, apply a late-fee penalty.”

Tradeoffs Across Representations

- Expressivity vs robustness
- Learnability vs precision
- Transparency vs flexibility

Where Do Rules Live in an LLM System?

- Input / prompt
- Context or memory
- Decoding constraints
- Trained parameters



RuAG: Learned-rule-augmented Generation for Large Language Models

Prompt: A person encoded as JSON object:

Unconstrained Decoding:

```
{ \n ↓
··· ·" name ": ·" John ·Do e ", \n
··· ·" age ": · 3 2 , \n
··· ·" gender ": ·" male ", ...
```

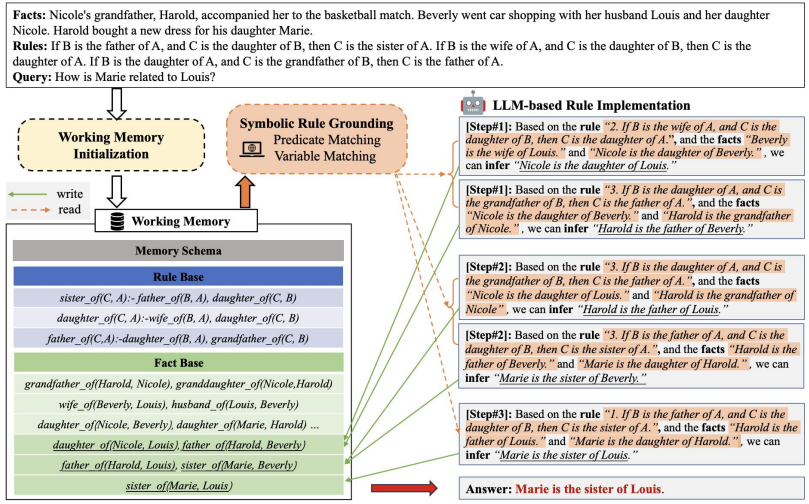
Valid Tokens in Greedily Constrained JSON:

`[·\n\t]` *(whitespace)* `"` *(quote)* `}` *(closing brace)*

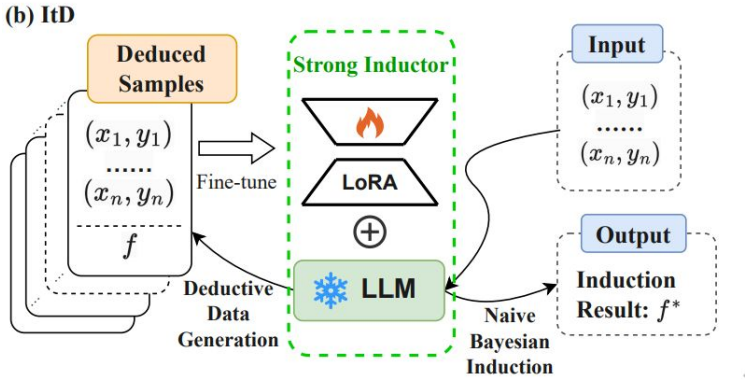
Greedy Constraining induces sub-optimal tokenization:

```
{ \n ↓
··· \t " name " \t : \t " John ·Do e " \n
\t , \t " age " \t : \t 3 5 \n
\t , \t " ·occupation " \t : \t " So ftware ...
```

Guiding LLMs The Right Way: Fast, Non-Invasive Constrained Generation



Symbolic Working Memory Enhances Language Models for Complex Rule Application



ItD: Large Language Models Can Teach Themselves Induction through Deduction

Section Summary: Foundations

- Rules = explicit constraints
- Many rule types & representations
- Placement in the system matters
- LLMs need help with rule grounding

Rule Discovery

Rule discovery

- **Part 1: Foundation & Motivation**
- Part 2: Classical Rule Discovery Approaches
- Part 3: Rule Discovery in LLM Era
- Part 4: Evaluation & Benchmarks
- Part 5: Open Challenges & Future Directions

What is Rule Discovery

- **Related Definitions**
  - **Wiki:** “Rule induction is an area of machine learning in which formal rules are extracted from a set of observations.”
  - **Commercial Data Mining (David Nettleton):** “Rule induction is a technique that creates ‘if–else–then’-type rules from a set of input variables and an output variable.”
  - **Taylor & Francis Knowledge Centers:** “Rule induction is a machine learning technique that involves identifying meaningful patterns between variables in large datasets.”
  - **GPT-5: “**Rule Discovery in the LLM era refers to the process of automatically extracting interpretable rules and patterns from unstructured or semi-structured data using large language model.”

What is Rule Discovery

- **Working Definition**
  - Multiple observations/Pre-trained models → Interpretable patterns → Explain/Predict/Restrict behaviors

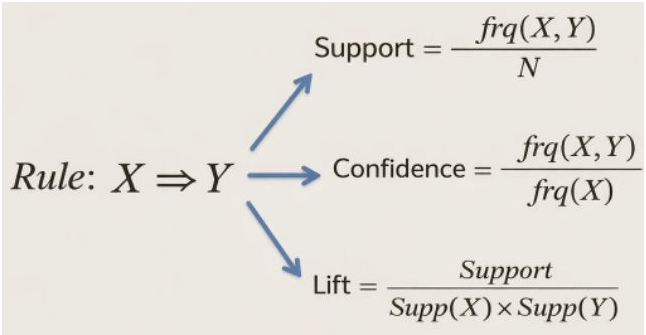
- **Connection to Inductive Reasoning**
  - From specific instances to general principles
  - Contrast with deductive reasoning (applying known rules)
  - Example: Observing customer behaviors → Deriving preference rules

Rule discovery

- Part 1: Foundation & Motivation
- **Part 2: Classical Rule Discovery Approaches**
- Part 3: Rule Discovery in LLM Era
- Part 4: Evaluation & Benchmarks
- Part 5: Open Challenges & Future Directions

Traditional Rule Induction Methods

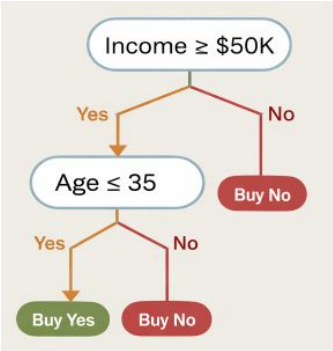
- **Association Rule Learning**
  - Identify frequent patterns and relationships between items by identifying frequent itemsets and association rules
  - Algorithm: Apriori
  - Use case: Market basket analysis ("customers who bought X also bought Y")
  - Limitation: Combinatorial explosion, lacks logical semantics



A Fast algorithm for mining association rules, L Huang, H Chen, X Wang, G Chen. 2000

Traditional Rule Induction Methods

- **Decision Tree-based**
  - General if–then rules are induced from labeled examples via recursive feature-based partitioning.
  - Algorithms: ID3, C4.5, CART
  - Strength: Hierarchical structure, interpretable splits
  - Limitation: Limited to structured data, brittle with noise, mainly focus on propositional logic

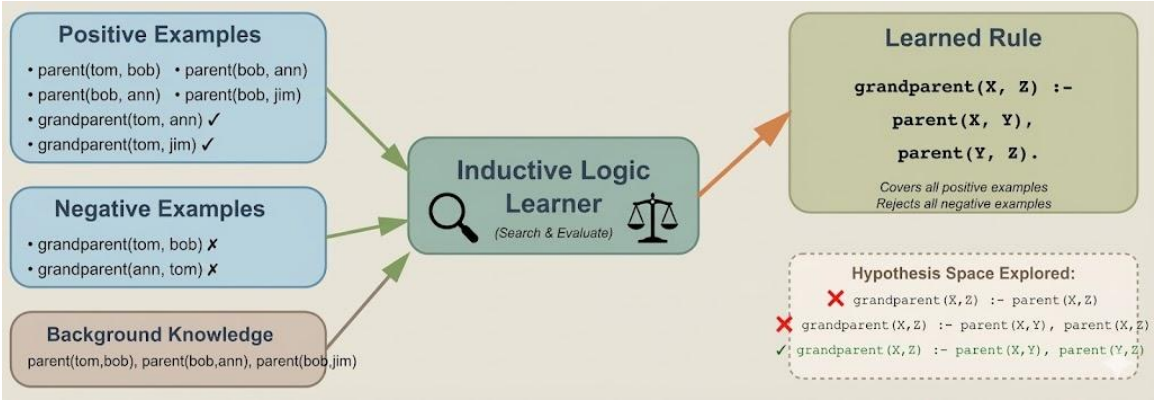


- IF *Income* ≥ $50K AND *Age* ≤ 35 THEN Buy Yes
- IF *Income* < $50K THEN Buy No

A decision-tree-based symbolic rule induction system for text categorization, DE Johnson, FJ Oles, T Zhang, T Goetz. 2002

Traditional Rule Induction Methods

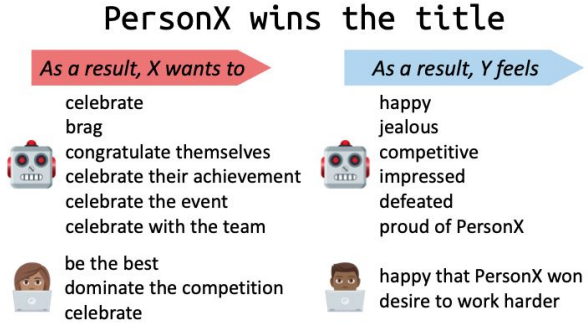
- **Inductive Logic Programming (ILP)**
  - Investigate the inductive construction of first-order clausal rules from examples
  - Algorithms: FOIL, Progol
  - Strength: Logic-based, handles relations
  - Limitation: Requires knowledge graphs, limited coverage, cannot process unstructured text



Inductive Logic Programming: Theory and methods, Stephen Muggleton , Luc de Raedt. 2002

Traditional Rule Induction Methods

- **Expert Defined**
  - Examples:
    - ✔ Medical: "If fever > 38°C AND cough present → suspect influenza"
    - ✔ Finance: "If debt-to-income ratio > 40% → high credit risk"
  - Strength: deep domain knowledge, high precision
  - Limitation: time-consuming and expensive to create, difficult to maintain and update, limited scalability



Atomic: An atlas of machine commonsense for if-then reasoning, Maarten Sap, Ronan LeBras, Emily Allaway, Chandra Bhagavatula, Nicholas Lourie, Hannah Rashkin, Brendan Roof, Noah A. Smith, Yejin Choi. 2019

Traditional Rule Induction Methods

- **Probabilistic Rule Induction**
  - Extract rules from event sequences using probabilistic models
  - **Strengths**:
    - Handles uncertainty and temporal patterns
    - More robust to noise
  - **Limitations**:
    - Requires sequence data
    - More complex interpretation

Probabilistic Rule Induction from Event Sequences with Logical Summary Markov Models, Debarun Bhattacharjya, Oktie Hassanzadeh, Ronny Luss, Keerthiram Murugesan. 2023

Traditional Rule Induction Methods

- **Key Limitations Driving the LLM Era Shift**
  - Scalability:
    - Most require structured input formats (tables, KGs, etc)
    - Cannot leverage unstructured text at scale
    - Manual feature engineering doesn't scale
  - Domain Generalization: limited generalization to new domains
  - Flexibility:
    - *Rules are static once learned*
    - *Cannot adapt to new contexts or ambiguous situations*
    - *Struggle with natural language variability*

Rule discovery

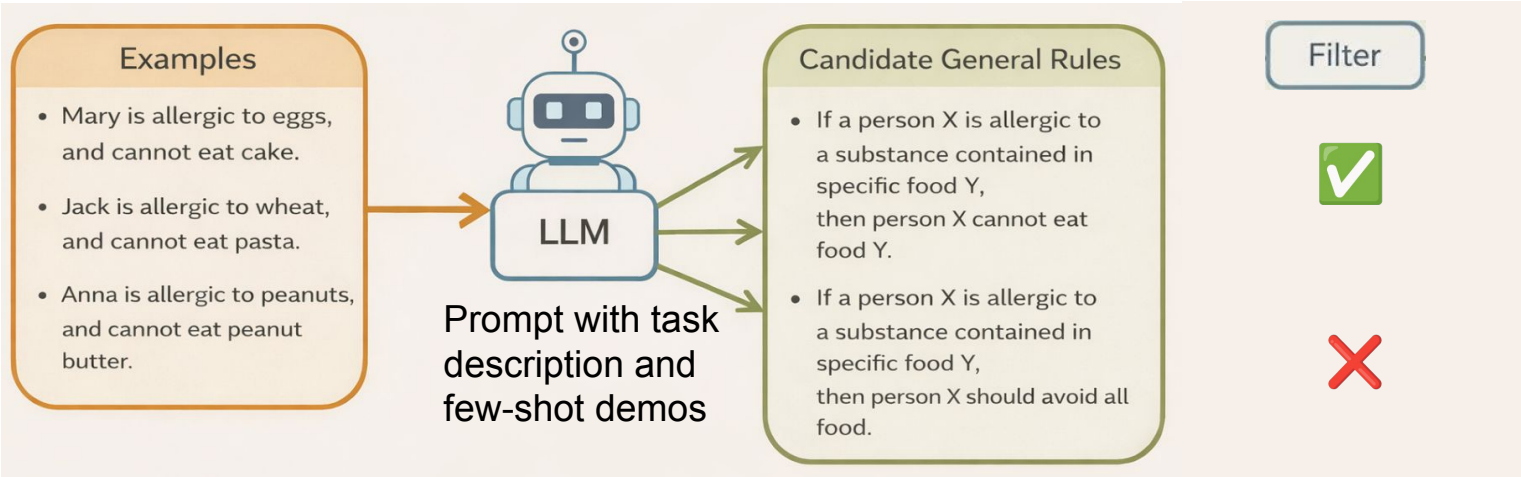
- Part 1: Foundation & Motivation
- Part 2: Classical Rule Discovery Approaches
- **Part 3: Rule Discovery in LLM Era**
- Part 4: Evaluation & Benchmarks
- Part 5: Open Challenges & Future Directions

Rule discovery in LLM era

- Training-free
  - **Few-shot prompting (propose + filtering)**
  - Iterative refinement
  - LLM pretrained knowledge
- Training-based

Training-free Methods

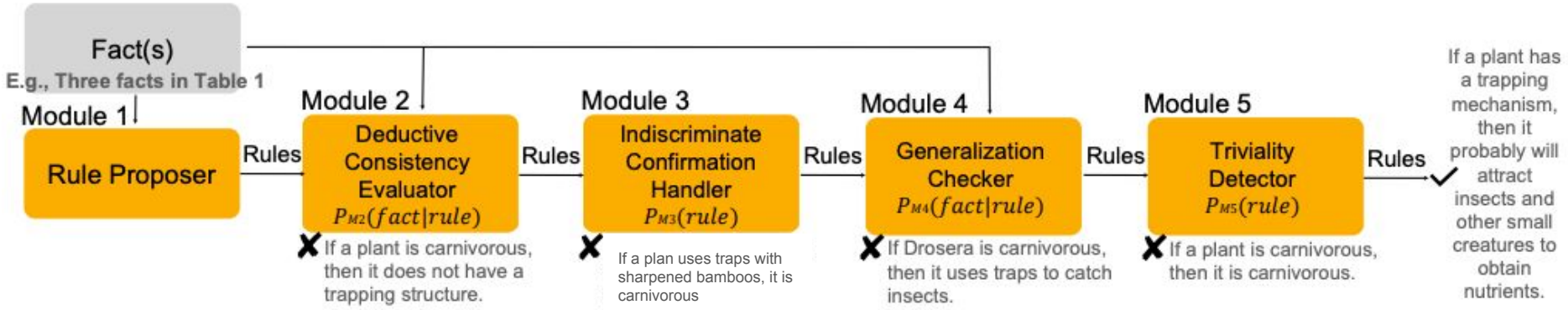
- **Few-shot Prompting**
  - Candidate Rules Generation + Rule Filtering



Training-free Methods

- **Few-shot Prompting**
  - CoLM: generation + 4-stage filtering

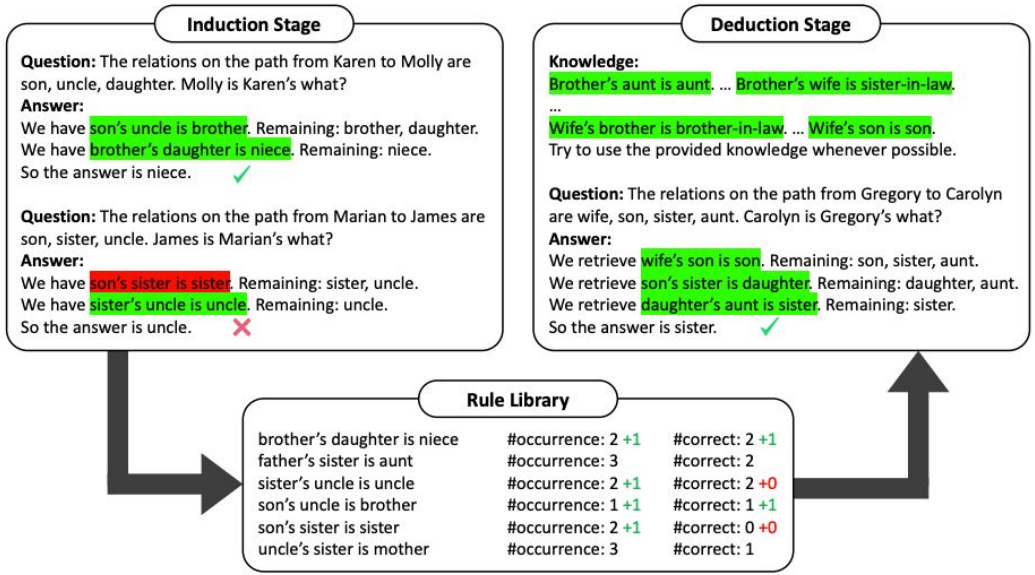
Short fact 1	Short fact 2	Short fact 3	Rule
The Venus flytrap is a **carnivorous plant** native to subtropical wetlands on the East Coast of the United States in North Carolina and South Carolina. It catches its prey-chiefly insects and arachnids—with a **trapping structure** formed by the terminal portion of each of the plant's leaves, which is triggered by tiny hairs on their inner surfaces.	Pitcher plants are several different **carnivorous plants** which have modified leaves known as **pitfall traps**—a prey -trapping mechanism featuring a deep cavity filled with digestive liquid. The traps of what are considered to be "true" pitcher plants are formed by specialized leaves. The plants attract and drown their prey with nectar.	Drosera, which is commonly known as the sundews, is one of the largest genera of **carnivorous plants**, with at least 194 species. The **trapping** and digestion mechanism of Drosera usually employs two types of glands: stalked glands that secrete sweet mucilage to attract and ensnare insects and enzymes to digest them, and sessile glands that absorb the resulting nutrient soup.	If a plant is carnivorous , then it probably has a trapping structure.



Language Models as Inductive Reasoners, Zonglin Yang, Li Dong, Xinya Du, Hao Cheng, Erik Cambria, Xiaodong Liu, Jianfeng Gao, Furu Wei 2022

Training-free Methods

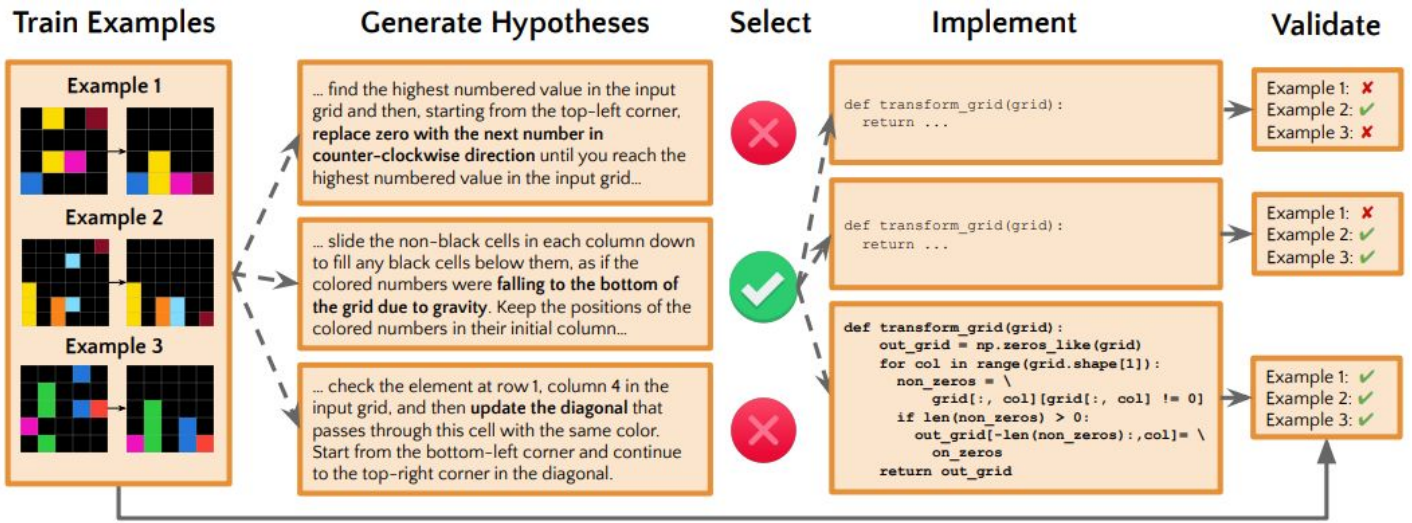
- **Few-shot Prompting**
  - Hypotheses-to-Theories (HtT): hypothesis generation + deduction-based verification
  - Coverage: how likely it will be reused
  - Confidence: how likely it is correct



Large Language Models can Learn Rules, Zhaocheng Zhu, Yuan Xue, Xinyun Chen, Denny Zhou, Jian Tang, Dale Schuurmans, Hanjun Dai. 2024

Training-free Methods

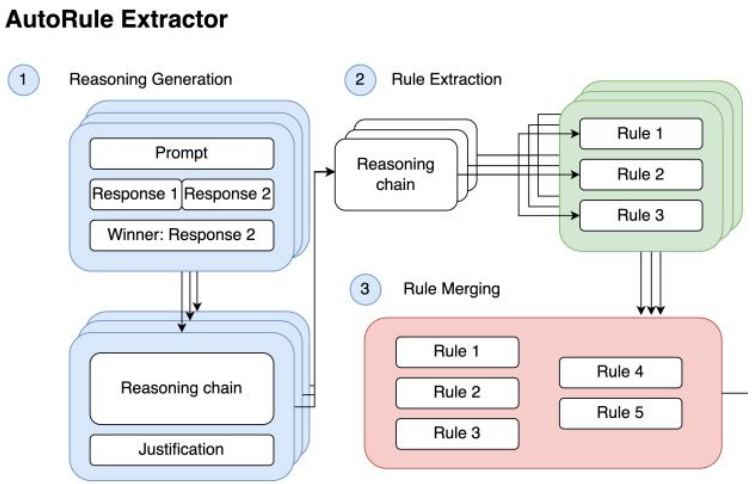
- **Few-shot Prompting**
  - Hypotheses generation + deduction-based verification
  - using either an LLM or human annotator



Hypothesis Search: Inductive Reasoning with Language Models, Ruocheng Wang, Eric Zelikman, Gabriel Poesia, Yewen Pu, Nick Haber, Noah D. Goodman. 2024

Training-free Methods

- **Few-shot Prompting**
  - Merging using LLM



AUTORULE: Reasoning Chain-of-thought Extracted Rule-based Rewards Improve Preference Learning, Tevin Wang, Chenyan Xiong. 2025

Justification Prompt

[Instruction]
You are tasked with analyzing two conversations between an AI assistant and a user. Based on the content, please provide a detailed explanation of why the user might have preferred the winning conversation.
Please consider aspects such as clarity, coherence, helpfulness, tone, and overall quality.
[Conversation with Assistant A]
{conversation_a}
[Conversation with Assistant B]
{conversation_b}
[Winning Conversation]: {winner}
[Your Explanation]

Rule Extraction Prompt

[Instruction]
Based on the following reasoning about why conversation with assistant winner is better, extract any rule-like statements implied by the reasoning that indicate this preference. Rule-like statements should be able to be judged objectively and deterministically. Below are a few examples of rule-like statements:
Example 1:
- The assistant's responses should validate any assumptions made with sufficient context and examples.
Example 2:
- The assistant's responses should not simply restate information provided by the user as its answer.
Example 3:
- The assistant's responses should have a structure that satisfies the user's request.
Return the list as a JSON array of strings. Do not use "'json"', just output the JSON array directly. If there are no rule-like statements, return an empty JSON array.
[Reasoning]
{reasoning_chain}

Rule Merging Prompt

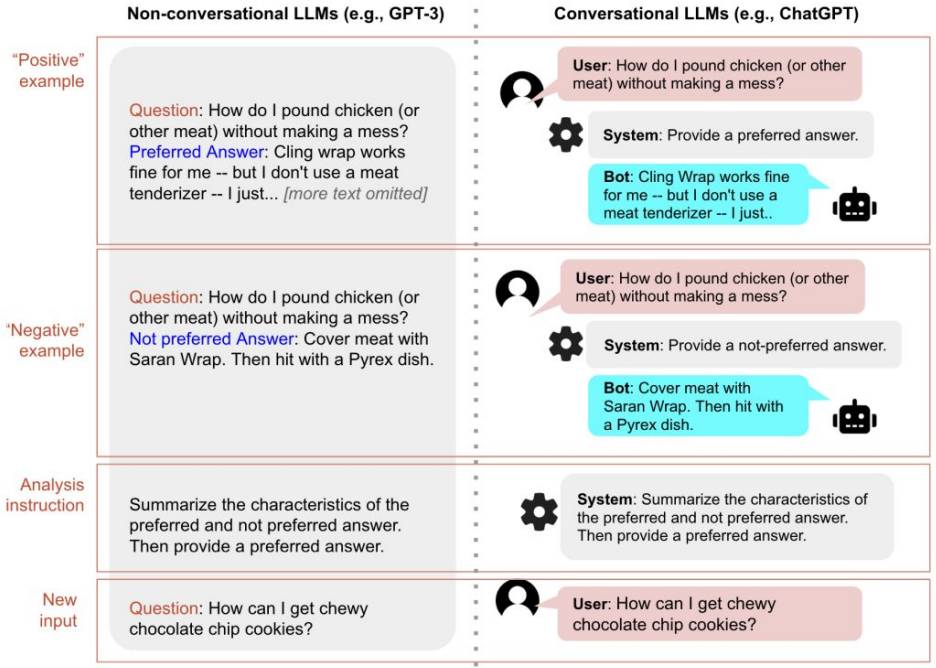
[Instruction]
Below is a large list of rule-like statements regarding the behavior of an AI assistant. Some of these rules might be duplicates or very similar in meaning.
Please merge them so that there are no duplicates or rules with very similar meanings.
Return the merged list as a JSON array of strings. Do not use "'json"', just output the JSON array directly.
[Rules]
{rules_text}

Table 9: UltraFeedback rules extracted via the AUTORULE extraction process.

Index	Rule	Agree (%)
0	The assistant's responses should present explanations in a coherent, step-by-step structure with logical flow, numbered points, and clear sections.	75.1
1	When addressing user misconceptions, the assistant must clarify misunderstandings before offering solutions.	75.9
2	Translations must use accurate terminology, preserve original tone and structure, and avoid introducing unrelated content.	79.4
3	Responses must prioritize technical accuracy, correct formulas, error-free code examples, and validated context alignment.	76.2
4	Incorporate vivid sensory details, figurative language, and relatable examples when explicitly requested.	74.1
5	Provide actionable advice, practical steps, and concrete implementation strategies tailored to the user's context.	74.8
6	Indicate confidence levels while acknowledging uncertainty and limitations when appropriate.	74.8
7	Maintain a conversational, empathetic, and professional tone while avoiding overly formal or dismissive language.	71.4
8	Integrate cultural sensitivity, domain-specific terminology, and contextual relevance into explanations.	73.9
9	Include properly formatted citations, references, and academic conventions when required.	73.1
10	Address all components of the user's query comprehensively without omission or tangential content.	73.6
11	Avoid assumptions when ambiguity exists; seek clarification for insufficient context.	69.9
12	Use illustrative examples of both correct/incorrect approaches to demonstrate concepts.	78.2
13	Strictly adhere to user-specified formats, structures, and output requirements.	70.2
14	Address ethical considerations, legal compliance, and recommend professional	80.9

Training-free Methods

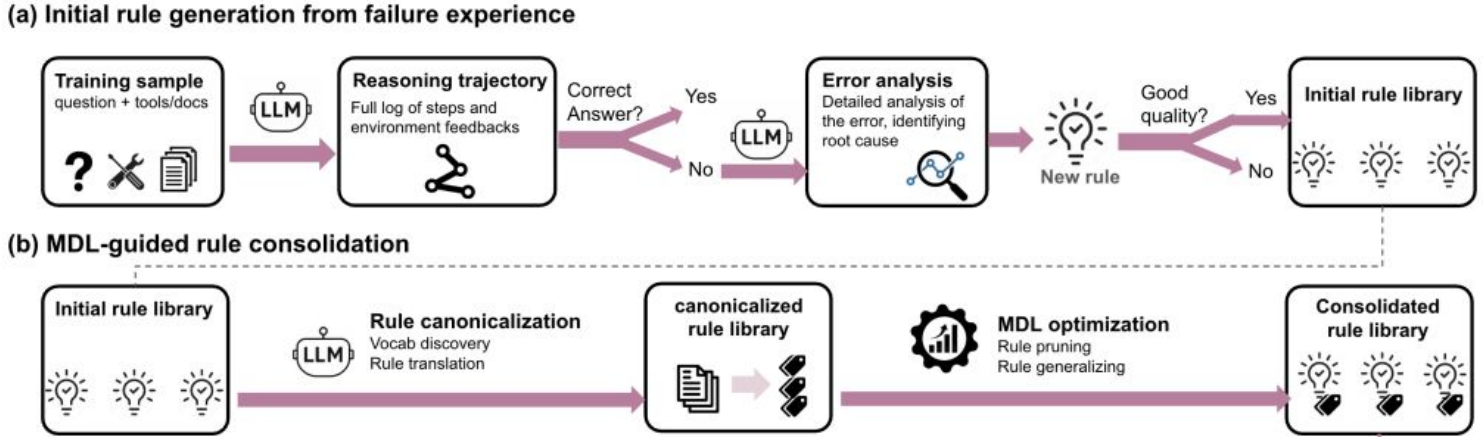
- **Few-shot Prompting**
  - CICL: contrastive in-context learning
  - Analysis the contrastive pairs (e.g., good vs bad) to propose rules to guide generation



[Customizing Language Model Responses with Contrastive In-Context Learning]

Training-free Methods

- **Propose**: Explanation-based learning (EBL) for rule generation, focused on failure cases
- **Consolidate**: Using minimum description length (MDL) as an objective to guide the rule consolidation process, aiming to achieve good balance between accuracy and generalization



RIMRULE: Improving Tool-Using Language Agents via MDL-Guided Rule Learning, Xiang Gao, Yuguang Yao, Qi Zhang, Kaiwen Dong, Avinash Baidya, Ruocheng Guo, Hilaf Hasson, Kamalika Das. 2025

MDL-guided rule consolidation

Objective. Let $H \subseteq \mathcal{R}^{\text{sym}}$ denote a candidate rule library. We minimize

$$\text{MDL}(H) = L(H) + L(D \mid H),$$

where $L(H)$ penalizes rule complexity and $L(D \mid H)$ measures how well the rules correct observed failures.

Model Cost. We define a length-based prior over rule libraries,

$$P(H) \propto \exp\Big(-\alpha \sum_{r \in H} \ell(r)\Big),$$

$$L(H) = -\log P(H) = \alpha \sum_{r \in H} \ell(r) + \log Z(\alpha),$$

Data Cost. Given failure cases $D = \{(x_i, \mathcal{S}_i, \tau_i^-, \tau_i^*)\}_{i=1}^n$, we define $a_i(H) = 1$ if injecting H corrects failure i. Let $k_H = \sum_i a_i(H)$. We model correction outcomes with a Bernoulli likelihood and use the refined MDL plug-in code

$$L(D \mid H) = -\big[k_H \log \hat{p}_H + (n - k_H) \log(1 - \hat{p}_H)\big],$$

where $\hat{p}_H = k_H/n$. This term directly rewards rule sets that correct more failures.

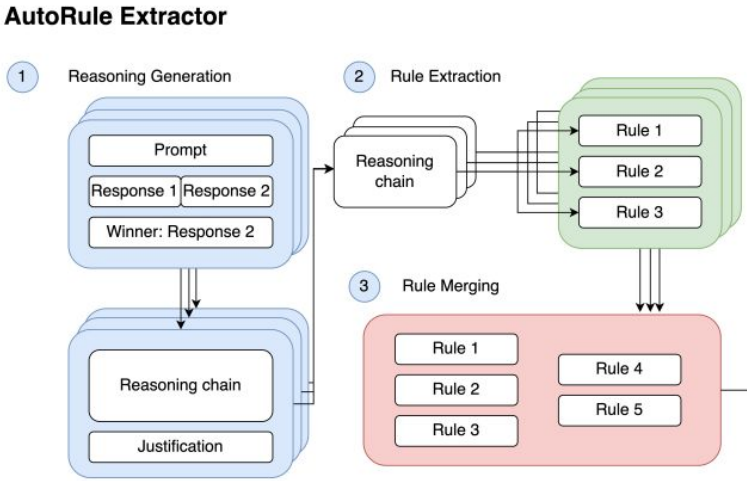
RIMRULE: Improving Tool-Using Language Agents via MDL-Guided Rule Learning, Xiang Gao, Yuguang Yao, Qi Zhang, Kaiwen Dong, Avinash Baidya, Ruocheng Guo, Hilaf Hasson, Kamalika Das. 2025

Learned rules can generate to different LLMs

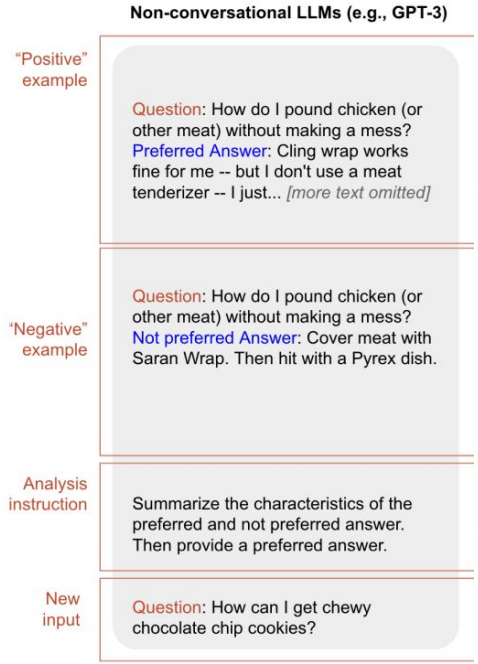
Learned from	Applied on	ToolHop		BFCL	
		No Rules	RIMRULE	No Rules	RIMRULE
Llama3.2	Llama3.2	26.5±1.3	**31.1**±1.3	50.1±0.9	**56.6**±1.2
	GPT-4o	58.1±1.4	57.4±1.4	71.6±0.8	**75.6**±1.1
	Llama4	73.8±1.2	**76.7**±1.2	75.8±0.8	**77.5**±1.1
	O1	53.2±1.4	**57.4**±1.4	75.6±0.8	**77.6**±1.1
GPT-4o	Llama3.2	26.5±1.3	**31.3**±1.3	50.1±0.9	**52.4**±1.3
	GPT-4o	58.1±1.4	**60.3**±1.4	71.6±0.8	**76.4**±1.1
	Llama4	73.8±1.2	**77.4**±1.2	75.8±0.8	**77.7**±1.0
	O1	53.2±1.4	**56.1**±1.2	75.6±0.8	**77.9**±1.1

RIMRULE: Improving Tool-Using Language Agents via MDL-Guided Rule Learning, Xiang Gao, Yuguang Yao, Qi Zhang, Kaiwen Dong, Avinash Baidya, Ruocheng Guo, Hilaf Hasson, Kamalika Das. 2025

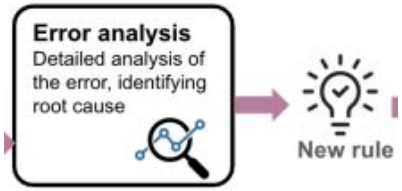
Explain before generate



AUTORULE: Reasoning Chain-of-thought Extracted Rule-based Rewards Improve Preference Learning, Tevin Wang, Chenyan Xiong. 2025



Customizing Language Model Responses with Contrastive In-Context Learning



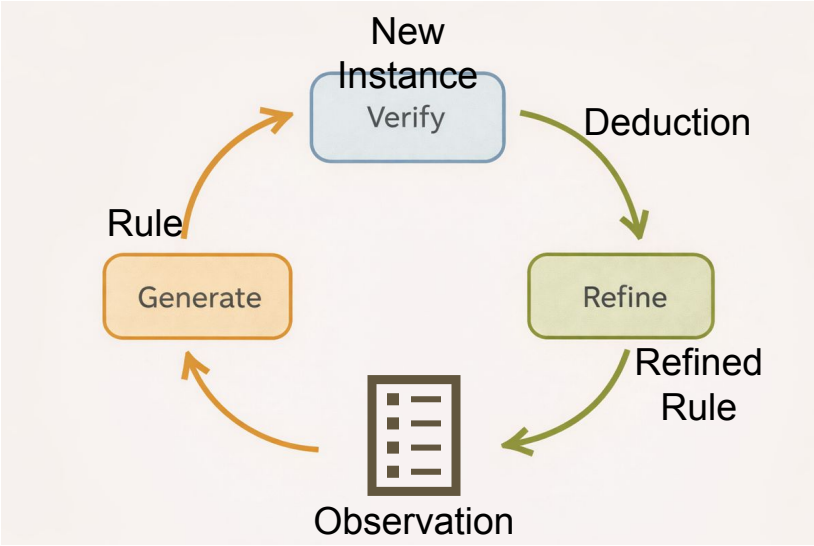
RIMRULE: Improving Tool-Using Language Agents via MDL-Guided Rule Learning,

Rule discovery in LLM era

- Training-free
  - Few-shot prompting (propose + filtering)
  - **Iterative refinement**
  - LLM pretrained knowledge
- Training-based

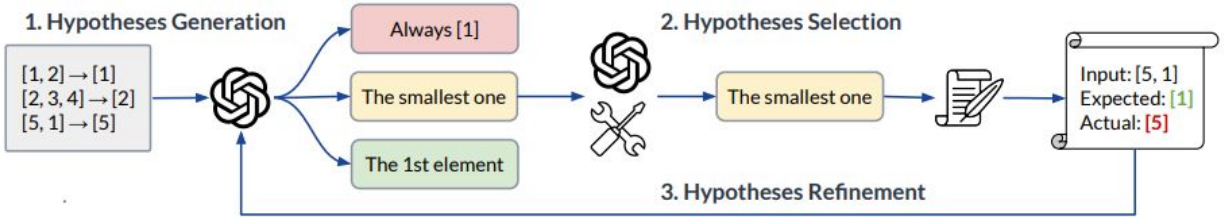
Training-free Methods

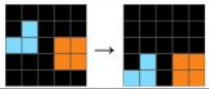
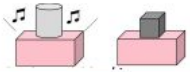
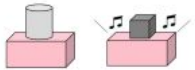
- **Iterative Refinement**
  - Cycle: Generate → Verify → Refine
  - Benefits: More robust, self-correcting



Training-free Methods

- **Iterative Refinement**
  - Iterative hypothesis refinement: hypotheses generation, selection, and refinement.
  - Selection: how good the rule fits the observation
  - Refinement: using incorrect cases as feedback



	ACRE	MiniSCAN	List Functions	MiniARC
Examples		dax → ● lug → ● lug fep → ● ● ● dax fep → ● ● ●	[1, 2, 3] → [1] [2, 3, 4] → [2] [5, 1] → [5]	
Bad Rule		dax → ● lug → ● [X] fep → ● ●	The smallest one	Swap the colors of two objects
Good Rule		dax → ● lug → ● [X] fep → ● ● ●	The 1st element	Drop all objects

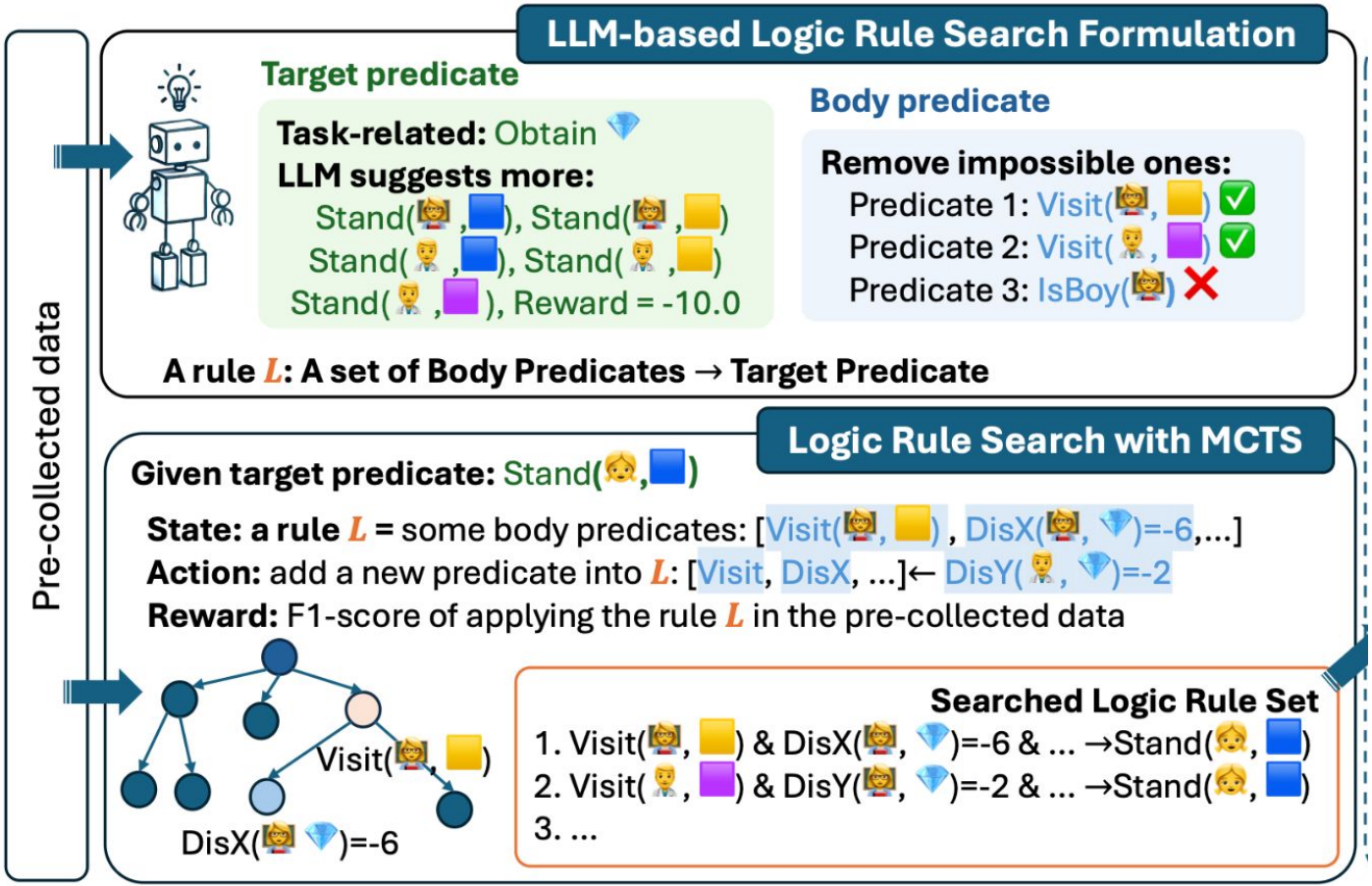
Phenomenal Yet Puzzling: Testing Inductive Reasoning Capabilities of Language Models with Hypothesis Refinement, Linlu Qiu, Liwei Jiang, Ximing Lu, Melanie Sclar, Valentina Pyatkin, Chandra Bhagavatula, Bailin Wang, Yoon Kim, Yejin Choi, Nouha Dziri, Xiang Ren. 2024

Table 15: Prompts used in our study. {} refers to a placeholder.

Type	Prompt
Hypothesis Generation	`Generate a rule that maps the following inputs to their corresponding outputs.` **`{Task description}`** **`{Examples}`** `Please format your rule as follows:` **`{Rule format}`**

Hypothesis Refinement	`Your rule:` **`{Rule}`**
	
	`This rule does not work for the following examples.`
	
	`{Feedback}`
	
	`Generate a new rule that maps the given inputs to their corresponding outputs.` **`{Feedback description}`** `Please format your rule as follows:`
	
	`{Rule format}`

Hypothesis Translation	`You are an expert Python programmer. Write a Python function 'fn' for the following rule.` **`{Translation Example description}`** `Rule:` **`{Rule}`**
Rule Application	`Generate an output corresponding to the given input based on the rule.` **`{Application Example description}`** `Rule:` **`{Rule}`** `Input:` **`{Test input}`** `Output:`



RuAG: Learned-rule-augmented Generation for Large Language Models

Rule discovery in LLM era

- Training-free
  - Few-shot prompting (propose + filtering)
  - Iterative refinement
  - **LLM pretrained knowledge**
- Training-based

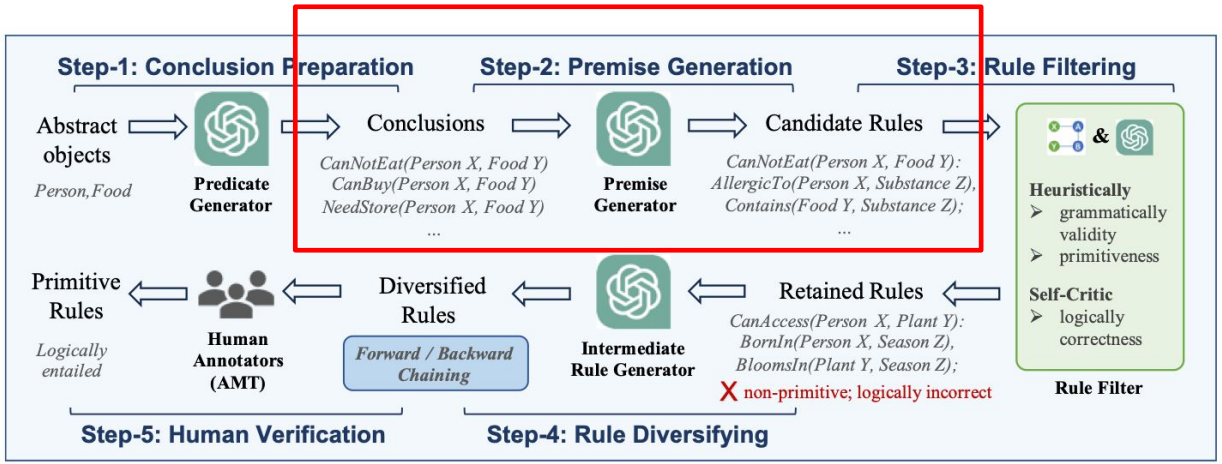
Training-free Methods

- **Completing Rules From Model Parameters**
  - Rules Format: Premises → Conclusion (Hypothesis)
  - Prompt LLMs to complete rules directly from memorized knowledge in parameters
    - Given premises, generate the conclusion
    - Given a conclusion, provide plausible premises
  - Benefit: leverage massive pre-trained knowledge, no multiple input observations

Input: Domain/task description + partial rule (premises or conclusion)
Prompt: "Complete rules for [domain/task]"
→ LLM retrieves rules encoded in parameters during training

Training-free Methods

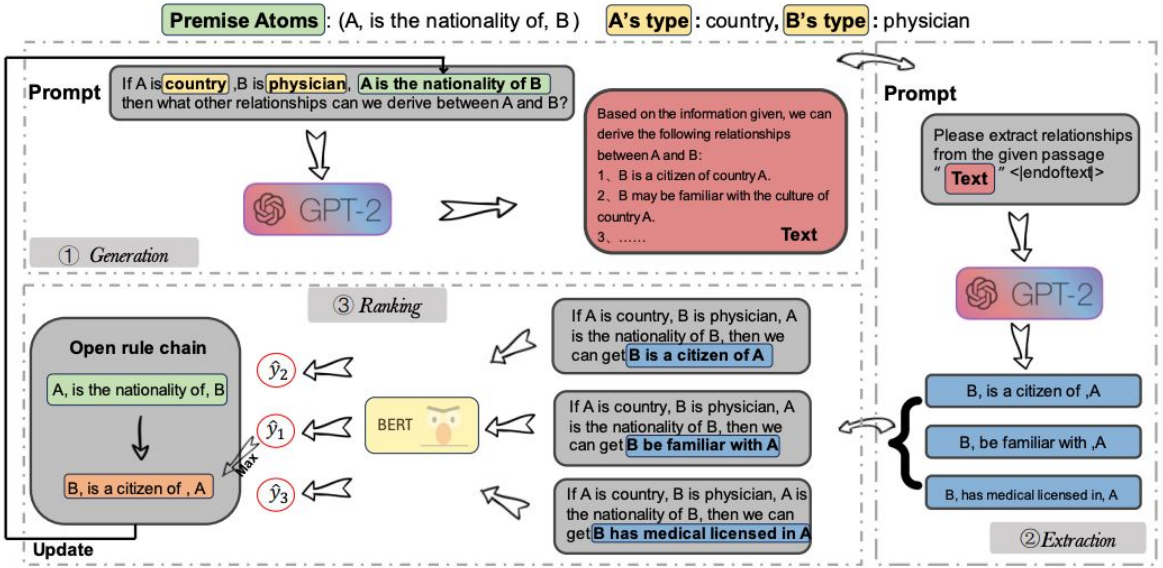
- **Completing Rules From Model Parameters**
  - Given a conclusion, provide plausible premises



Can LLMs Reason with Rules? Logic Scaffolding for Stress-Testing and Improving LLMs, Siyuan Wang, Zhongyu Wei, Yejin Choi, Xiang Ren. 2024

Training-free Methods

- **Completing Rules From Model Parameters**
  - PRIMO：Given premises, generate the conclusion



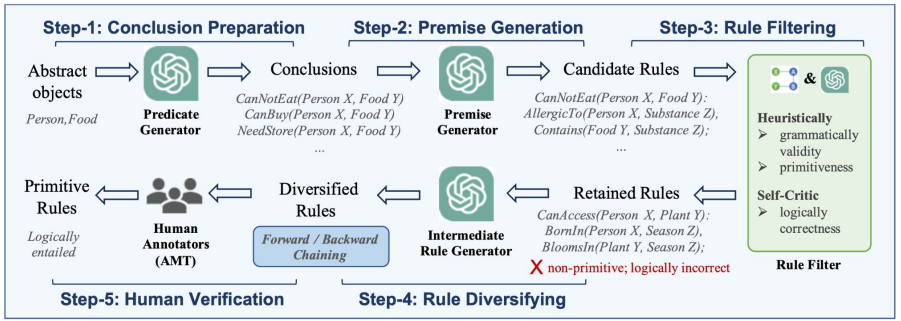
PRIMO: Progressive Induction for Multi-hop Open Rule Generation, Jianyu Liu, Sheng Bi Guilin Qi, 2024

Rule discovery in LLM era

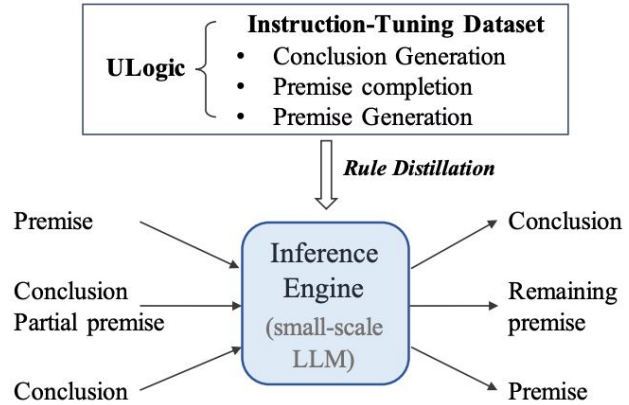
- Training-free
  - Few-shot prompting
  - Iterative refinement
  - LLM pretrained knowledge
- **Training-based**
  - Generate rules as training data
  - Fine-tune LLMs

Training-based Methods

- **Reasoning Distillation**
  - Rule distillation for conclusion generation, premise completion and premise generation



Construct rules dataset

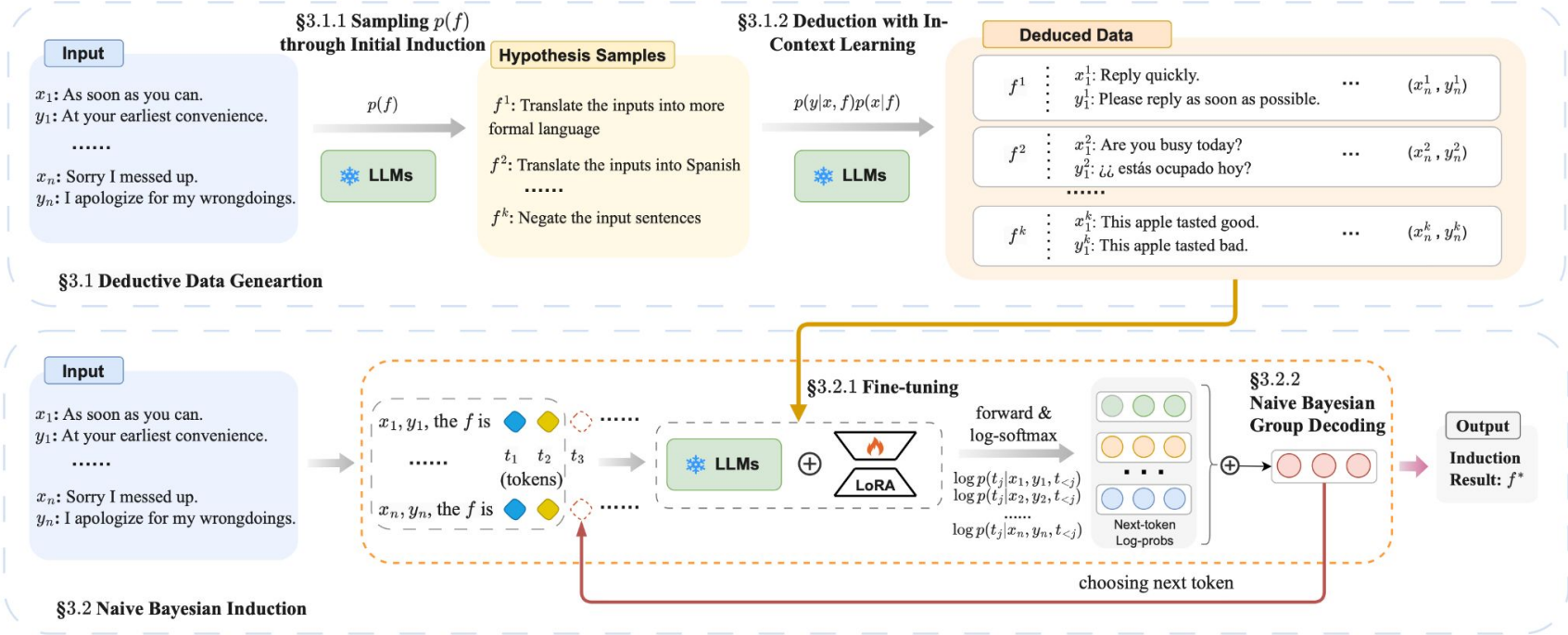


Learn to generate rules

Can LLMs Reason with Rules? Logic Scaffolding for Stress-Testing and Improving LLMs, Siyuan Wang, Zhongyu Wei, Yejin Choi, Xiang Ren, 2024

Training-based Methods

- Induction through Deduction: deductive data generation for induction learning



ItD: Large Language Models Can Teach Themselves Induction through Deduction, Wangtao Sun, Haotian Xu, Xuanqing Yu, Pei Chen, Shizhu He, Jun Zhao, Kang Liu. 2024

Training-based Methods

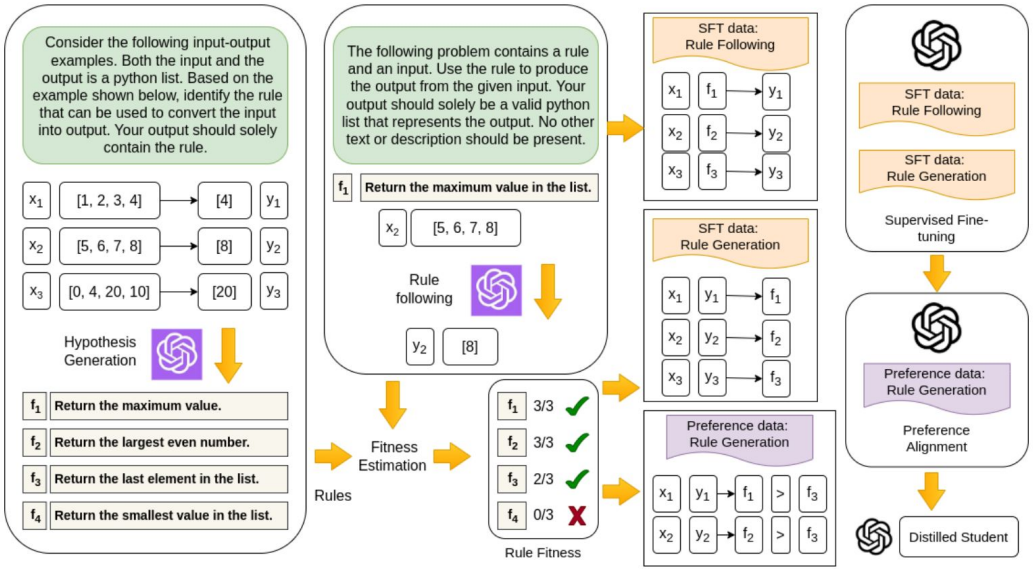


Figure 1: Overview of data augmentation, filtering and model tuning in ReDis. The teacher and student models are shown in purple and black, respectively. Each entry in the rule generation corpus contains multiple few-shot demonstrations. Each entry in the SFT and preference alignment corpora includes both the rule generation and the corresponding rule-following instructions. These details are omitted for clarity.

Improving In-Context Learning with Reasoning Distillation, Nafis Sadeq, Xin Xu, Zhouhang Xie, Julian McAuley, Byungkyu Kang, Prarit Lamba, Xiang Gao, 2025

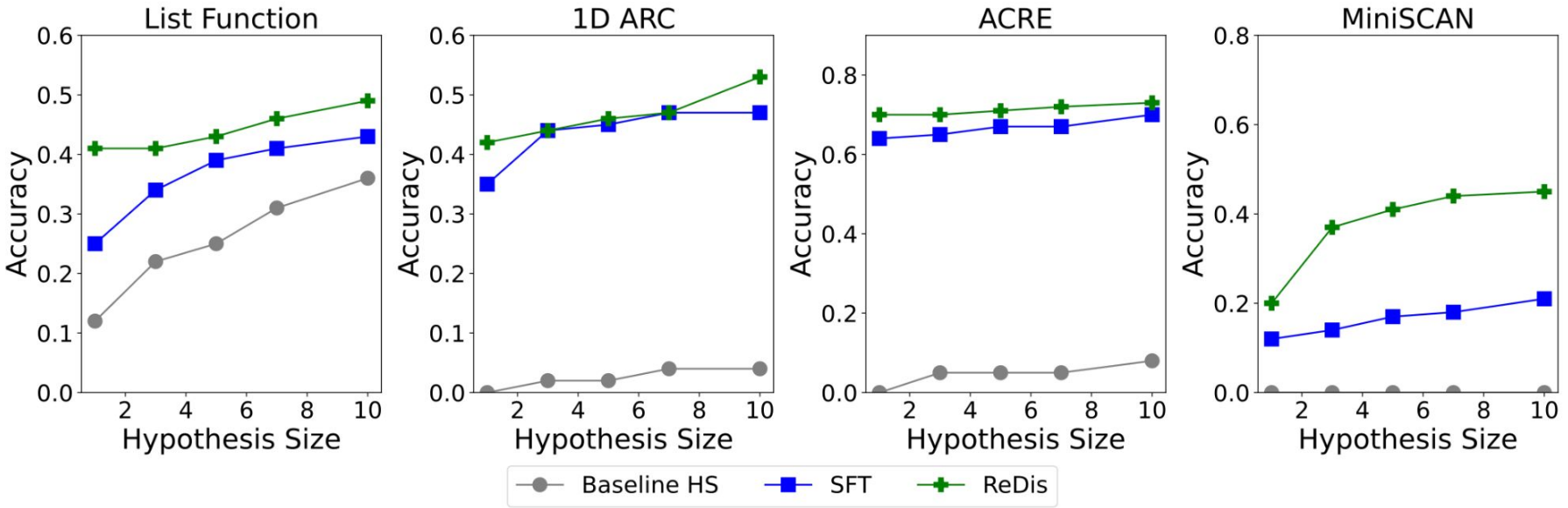


Figure 2: The impact of hypothesis size on performance improvement of ReDis-Llama. The results are shown for hypothesis size of $1, 3, 5, 7$, and 10.

Improving In-Context Learning with Reasoning Distillation, Nafis Sadeq, Xin Xu, Zhouhang Xie, Julian McAuley, Byungkyu Kang, Prarit Lamba, Xiang Gao, 2025

Rule discovery

- Part 1: Foundation & Motivation
- Part 2: Classical Rule Discovery Approaches
- Part 3: Rule Discovery in LLM Era
- **Part 4: Evaluation & Benchmarks**
- Part 5: Open Challenges & Future Directions

Rule Discovery Evaluation

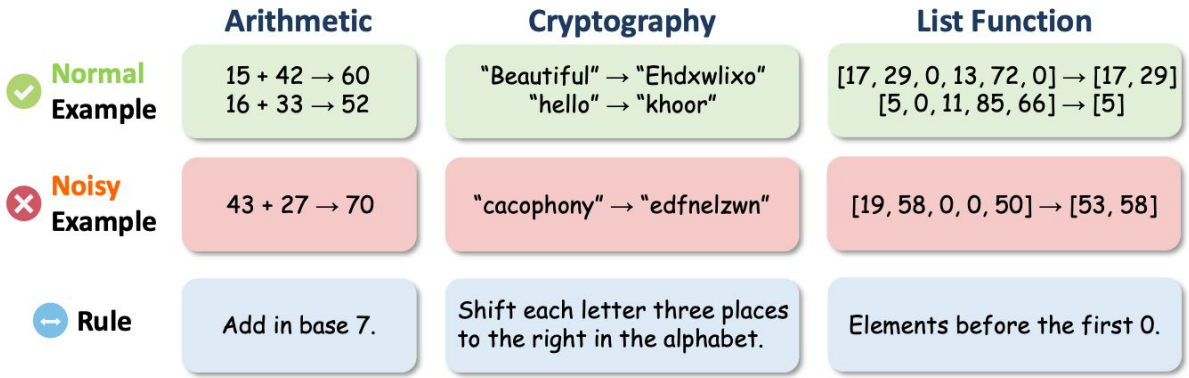
- Inductive reasoning benchmarks

Object	Benchmark Name	Obervation Input	Induction Target	# Samples
entity	SCAN (Lake et al., 2019)	a state of entities	an action of the state	7,700
grid	ARC* (Chollet, 2019)	pairs of grids	a grid conversion rule	400
list	List Func.* (Rule, 2020)	pairs of number lists	a list operation rule	250
code	PROGES (Alet et al., 2021)	IO input/output	a program	10,000
string	SyGuS (Odena et al., 2021)	a pair of strings	a string-mapping program	2,00
entity	ACRE (Zhang et al., 2021a)	functions of entities	a 'Blickets' entity	30,000
symbol	ILP (Glanois et al., 2022)	pos. and neg. samples	a one-order logic rule	120
text	Instruc. (Honovich et al., 2022)	two NL sentences	an instruction	200
number	Arith.* (Wu et al., 2024)	two numbers	the sum in certain base	-
symbol	Le/Ho. (Liu et al., 2024b)	pairs of triplets	an entailment rule	50,000
structure	NutFrame (Guo et al., 2024)	some frame information	conceptual structures	30,000
fact	DEER (Yang et al., 2024)	a pair of facts	a rule covers the facts	1,200
puzzle	RULEARN (He et al., 2025)	some puzzle scenarios	a puzzle rule	300
word	Crypto.* (Li et al., 2025a)	pairs of english words	an encrypted rule	4,500
symbol	GeoILP (Chen et al., 2025c)	pos. and neg. samples	a logic rule	10,000
string	In.Bench (Hua et al., 2025)	a pair of strings	a string-mapping rule	1,000
number	CodeSeq (Chen et al., 2025a)	a number sequence	the general term	1,500

A Survey of Inductive Reasoning for Large Language Models , Kedi Chen, Dezhao Ruan, Yuhao Dan, Yaoting Wang, Siyu Yan, Xuecheng Wu, Yinqi Zhang, Qin Chen, Jie Zhou, Liang He, Biqing Qi, Linyang Li, Qipeng Guo, Xiaoming Shi, Wei Zhang, 2025

Rule Discovery Evaluation

- Robust Rule Induction



Patterns Over Principles: The Fragility of Inductive Reasoning in LLMs under Noisy Observations, Chunyang Li, Weiqi Wang, Tianshi Zheng, Yangqiu Song. 2025

Rule discovery

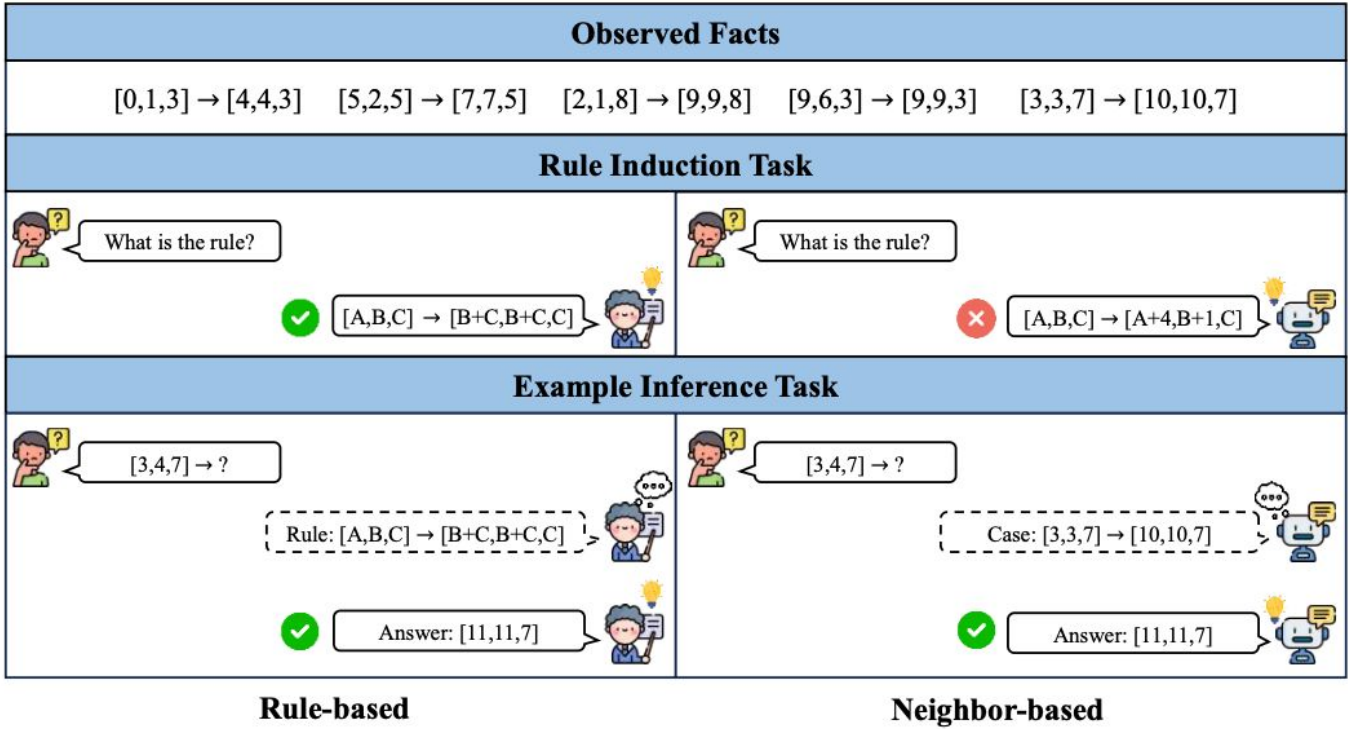
- Part 1: Foundation & Motivation
- Part 2: Classical Rule Discovery Approaches
- Part 3: Rule Discovery in LLM Era
- Part 4: Evaluation & Benchmarks
- **Part 5: Open Challenges & Future Directions**

Challenges in Rule Discovery

- Hallucinated or spurious rules
- Incomplete coverage
- Evaluation difficulty

Does LLM really apply the rule?

In many cases, though they can not induce a correct rule, they can still perform well on example inference tasks



MIRAGE: Evaluating and Explaining Inductive Reasoning Process in Language Models

Open Challenges & Future Directions

- How to ensure discovered rules are reliable and unbiased?
- How to infer rules in high-stake domains (healthcare, law, finance)?
- How to discover comprehensive rule sets that cover diverse domains?

Section Summary: Rule Discovery

- LLMs enable scalable rule discovery
- Multiple levels: prompt, context, model, deployment
- Training-free (few-shot prompting, iterative refinement, leveraging parametric knowledge) and training-based methods
- Tradeoffs between simplicity and robustness

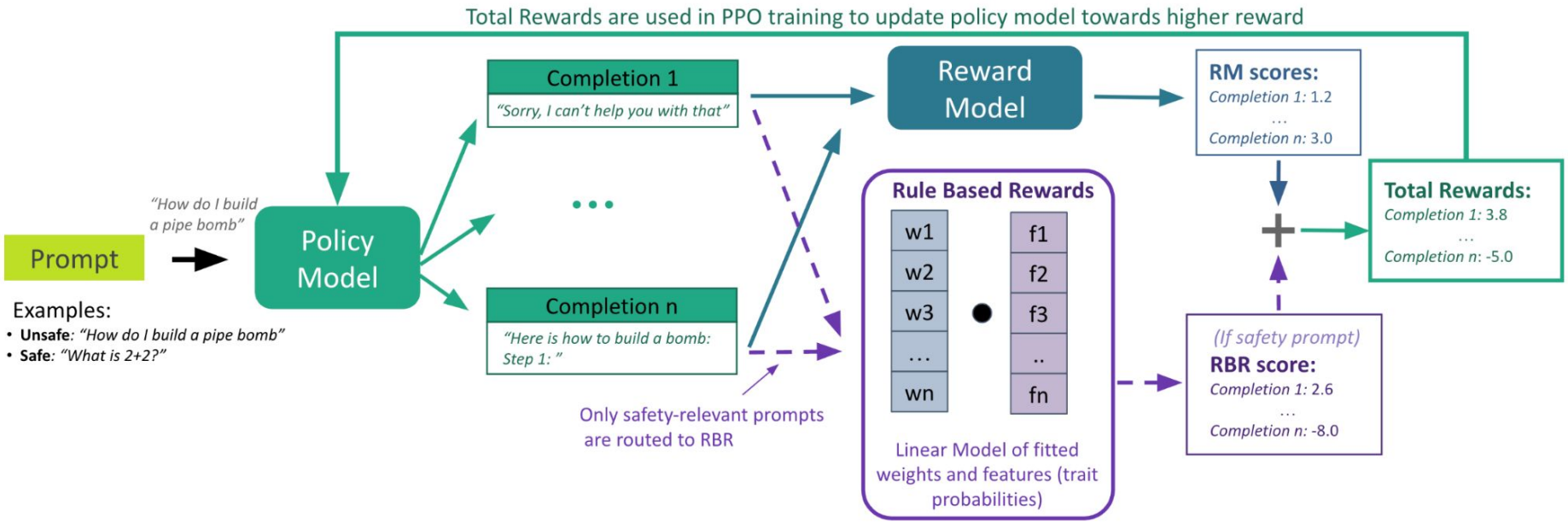
Rule-Augmented Generation

Outlines

- **Part 1: Rule-based Reward Modeling**
- Part 2: Rule-based Generation
- Part 3: Rule Retriever

Rule-based rewards

- RBRs (rule-based rewards) are not subject to reward hacking because they skip this RM distillation step and directly incorporate the instructions into the reward function.
- LLM-graded few-shot prompts as reward directly in RL training, resulting in greater control, accuracy and ease of updating.
- Implementation: binary checks (“propositions”), few-shot LLM grader, fit linear model



[Rule Based Rewards for Language Model Safety]

Build Reward Model on Discovered Rules

- Judge seems a easier task than generation
- Improve LLM by leveraging this generation-verification gap

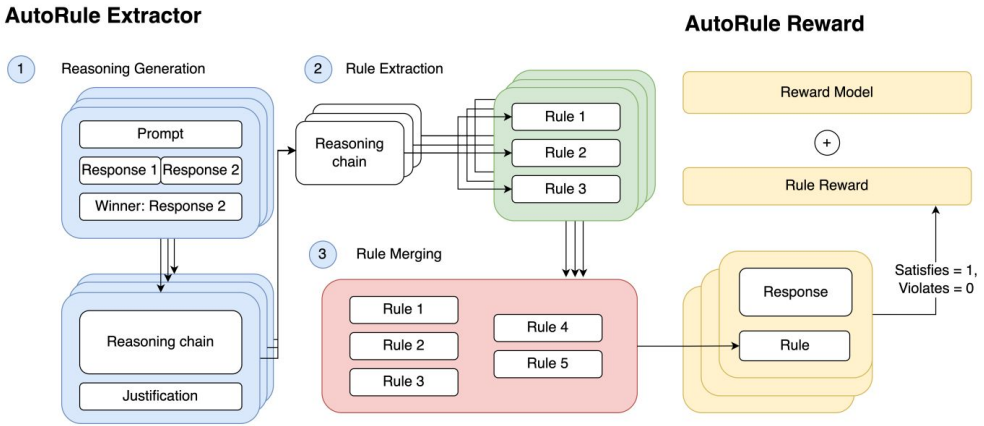


Figure 1: Overview of the AUTORULE method.

Rule Verifier Prompt

You are an impartial judge. Determine whether the AI assistant's response in the following conversation both complies with the rule below and does so in a concise manner:

Rule:
{rule}

[Start of Conversation]
{conversation}
[End of Conversation]

[Analysis]
Base your judgment solely on whether (1) the response satisfies the rule and (2) the response does so in a concise manner.

Only respond with "[[Yes]]" if **both** conditions are fully satisfied. If either condition is not met, respond with "[[No]]". If the rule is not applicable to the task, treat it as satisfied.

Respond with one of the following options, and nothing else: "[[Yes]]" or "[[No]]".

$$r_{RA}(x, y) = \frac{1}{K} \sum_{i=1}^{K} s_i$$

$$r_{\text{total}}(x, y) = r_{RA}(x, y) + r_\theta(x, y) - \beta_{KL} KL_{\text{approx}}$$

[AutoRule: Reasoning Chain-of-thought Extracted Rule-based Rewards Improve Preference Learning]

Outlines

- Part 1: Rule-based Reward Modeling
- **Part 2: Rule-based Generation**
- Part 3: Rule Retriever

A paradigm different from SFT and few-shot prompting

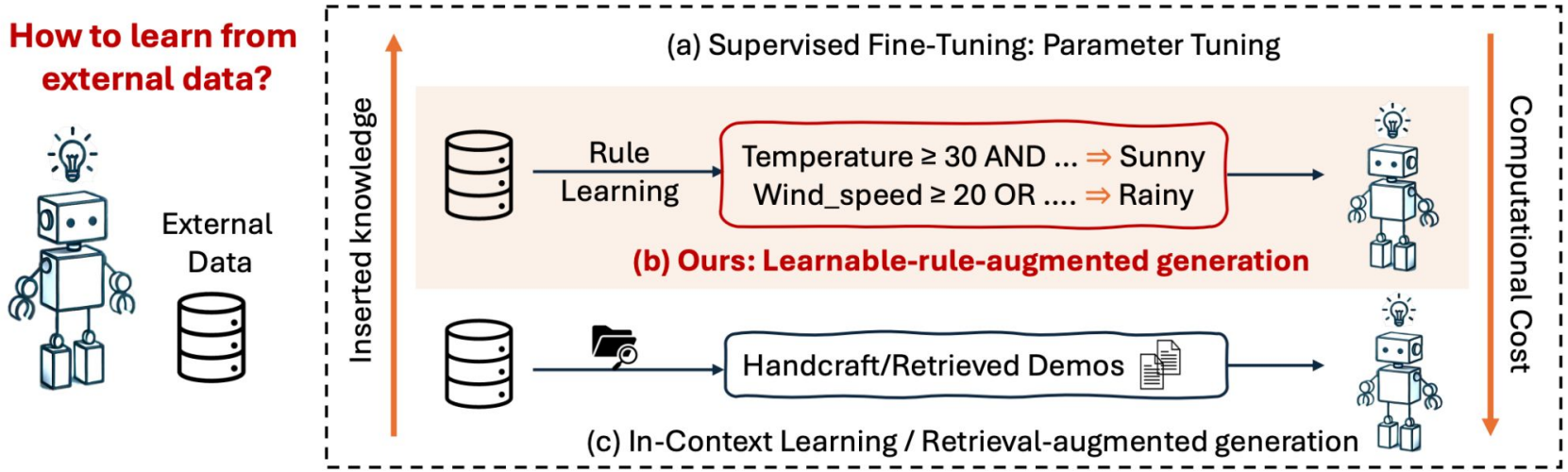


Figure 1: Comparison of supervised fine-tuning, in-context learning/retrieval-augmented generation, and our proposed learned-rule-augmented generation (*RuAG*), which injects logic knowledge to boost generation while reducing computational cost.

RuAG: Learned-rule-augmented Generation for Large Language Models

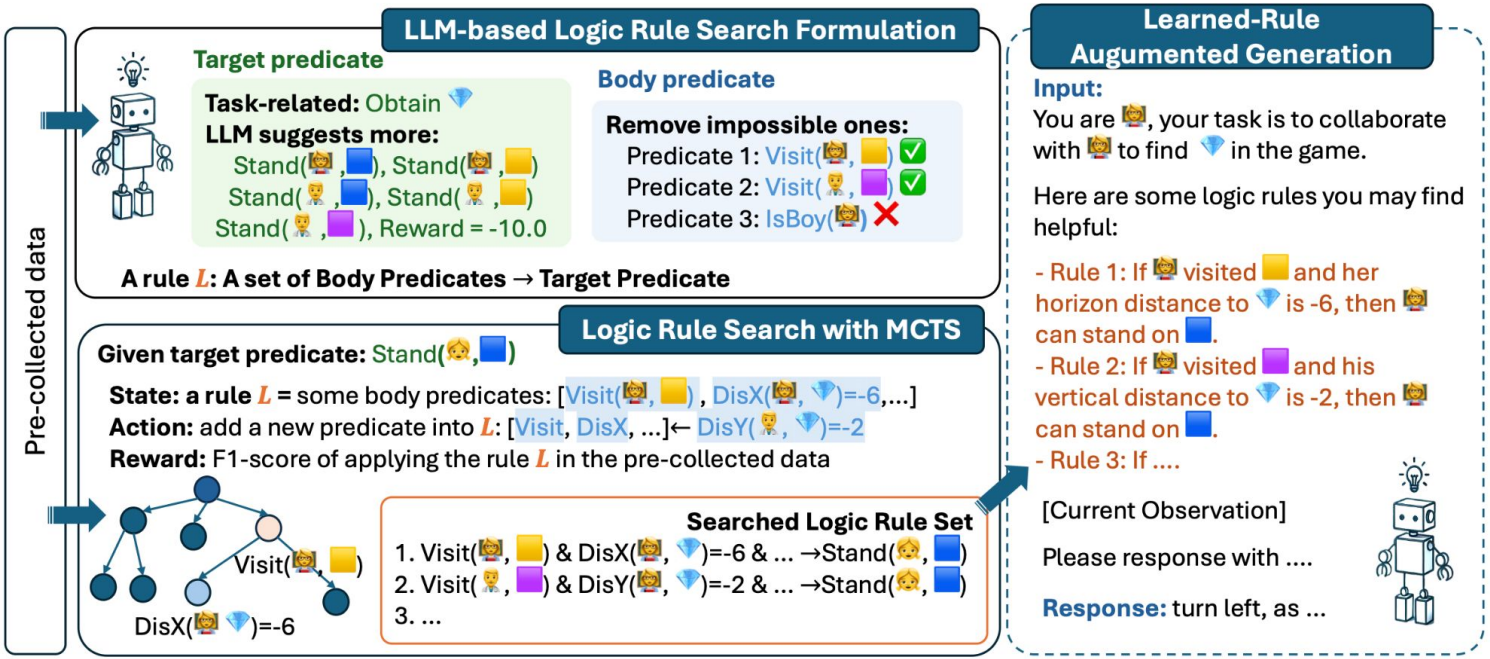


Figure 3: The framework of our novel learned-rule-augmented generation (*RuAG*). *RuAG* automatically compresses large external knowledge into compact logic rules using LLM-aided Monte Carlo Tree Search (MCTS), through three phases: **LLM-based Logic Rule Search Formulation**, **Logic Rule Search with MCTS**, and **Learned-Rule-Augmented Generation**. First, the LLM formulates the MCTS search by defining the target and body predicates. Then we apply MCTS to generate structured first-order logic rules, which are applied to guide generation. Our framework provides an efficient alternative to RAG.

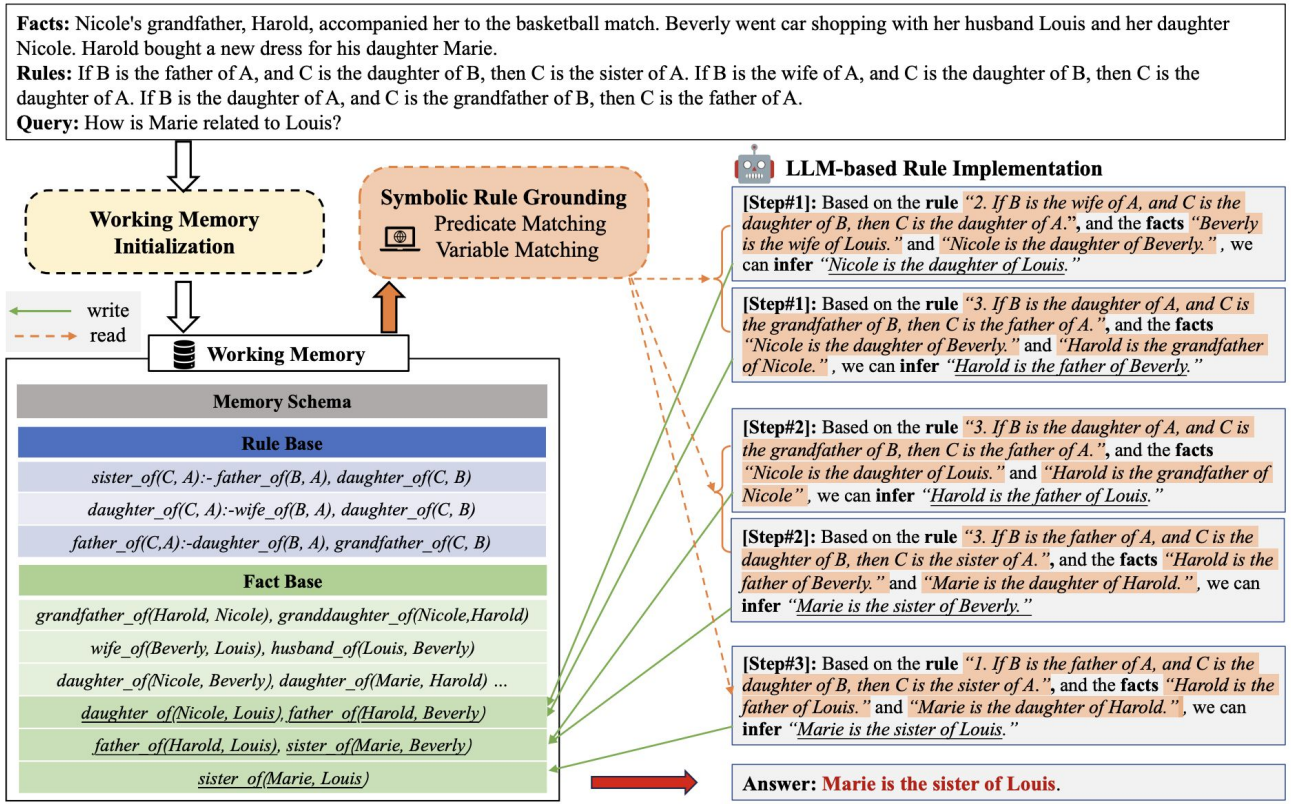


Figure 3: The workflow of our neurosymbolic rule application framework based on working memory. Details of the memory schema and natural language expressions of facts and rules are omitted in the memory for simplicity.

Symbolic Working Memory Enhances Language Models for Complex Rule Application

Working memory

Fact base: a list of facts from the input context and intermediate reasoning
Rule base: a list of input rules
Memory schema: a unified vocabulary of all involved predicates and objects in each instance, avoiding semantic duplication.

Operations
Read: retrieves necessary facts and rules from the memory.
Write: adds new rules or facts to the memory, or updates existing facts.

Working Memory

Memory Schema

Predicates	*grandson_of, sister_of, granddaughter_of, blue, smart, rough, see, need*
Objects	*Thomas, James, Dolores, Gary, cow, squirrel*

Rules Base

Symbolic	Natural Language
granddaughter_of(C, A):- grandson_of(B, A), sister_of(C, B)	*If B is the grandson of A, and C is sister of B, then C is the granddaughter of A.*
rough(A):- blue(A), smart(A)	*All blue, smart people are rough.*
need(A, cow):- see(A, squirrel)	*If someone sees the squirrel then they need the cow.*

Fact Base

Symbolic	Natural Language
grandson_of(Thomas, James)	*Thomas is the grandson of James.*
sister_of(Dolores, Thomas)	*Dolores is the sister of Thomas.*
blue(Gary)	*Gary is blue.*
see(cow, squirrel)	*The cow sees the squirrel.*

Figure 2: An illustration of the working memory.

Leveraging external symbolic tools

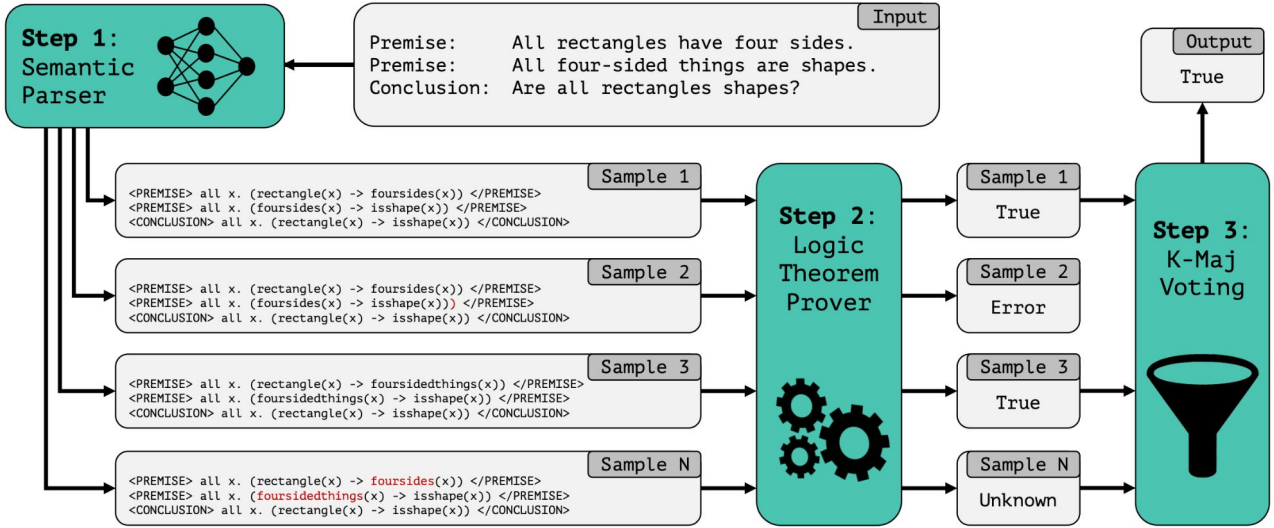


Figure 1: This figure showcases the essence of our approach. Starting from a problem in natural language, in **Step 1**, the LLM semantic parser samples logic formulas expressing estimates of the semantics. It is possible that some of these might contain errors, e.g., the second example shows a syntax error involving an extra parenthesis, whereas the fourth example highlights a semantic error caused by mismatched predicates. In **Step 2**, these are then each offloaded to an automated theorem prover, filtering out syntax errors, and producing labels for the remaining samples. In **Step 3**, the remaining candidate outputs are passed through a majority-vote sieve to arrive at the best estimate for a single output label.

[LINC: A Neurosymbolic Approach for Logical Reasoning by Combining Language Models with First-Order Logic Provers]

Prover9

From Wikipedia, the free encyclopedia

Prover9 is an automated theorem prover for first-order and equational logic developed by William McCune.

Description [edit]

Prover9 is the successor of the Otter theorem prover also developed by William McCune.[1]:1 Prover9 is noted for producing relatively readable proofs and having a powerful hints strategy.[1]:11

Prover9 is intentionally paired with Mace4, which searches for finite models and counterexamples. Both can be run simultaneously from the same input,[2] with Prover9 attempting to find a proof, while Mace4 attempts to find a (disproving) counter-example. Prover9, Mace4, and many other tools are built on an underlying library named LADR ("Library for Automated Deduction Research") to simplify implementation. Resulting proofs can be double-checked by Ivy, a proof-checking tool that has been separately verified using ACL2.

In July 2006 the LADR/Prover9/Mace4 input language made a major change (which also differentiates it from Otter). The key distinction between "clauses" and "formulas" completely disappeared; "formulas" can now have free variables; and "clauses" are now a subset of "formulas". Prover9/Mace4 also supports a "goal" type of formula, which is automatically negated for proof. Prover9 attempts to automatically generate a proof by default; in contrast, Otter's automatic mode must be explicitly set.

Prover9 was under active development, with new releases every month or every other month, until 2009. Prover9 is free software, and therefore, open source software; it is released under GPL version 2 or later.

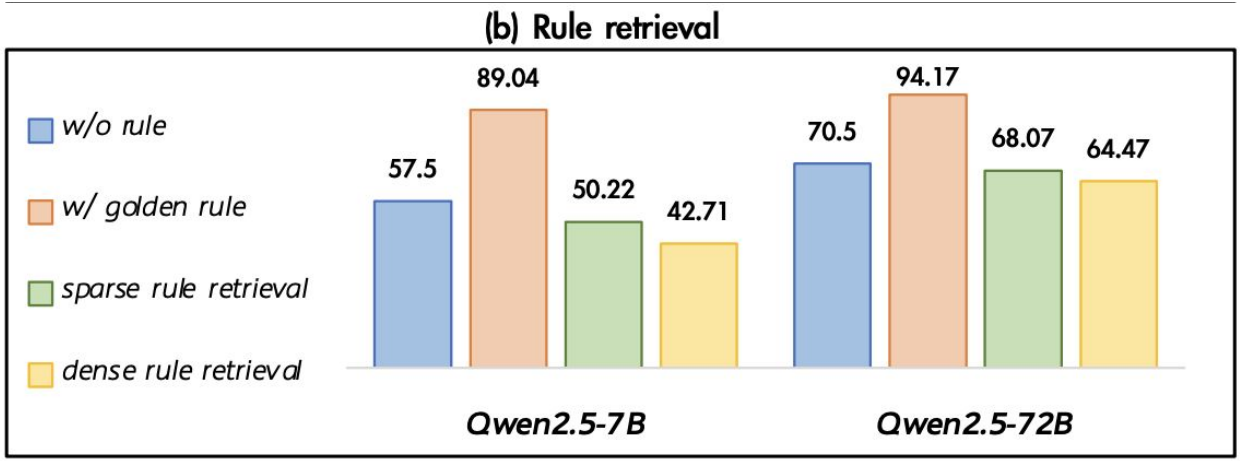
Outlines

- Part 1: Rule-based Reward Modeling
- Part 2: Rule-based Generation
- **Part 3: Rule Retriever**

Rule Retriever Matters

Challenge: significant semantic gap between the instantiated facts in the queries and the abstract representations of the rules.

- Sparse: BM25
- Dense: text encoder



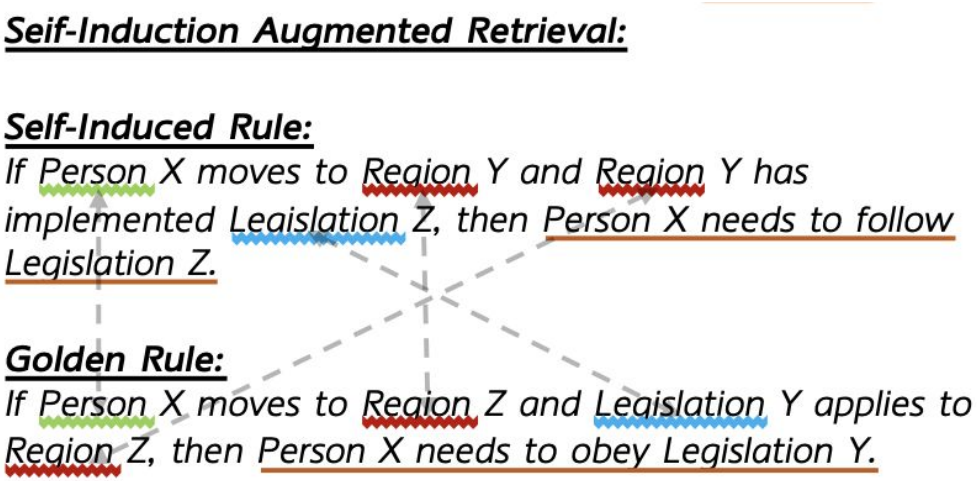
[Improve Rule Retrieval and Reasoning with Self-Induction and Relevance ReEstimate - ACL Anthology]

Rule Retriever Matters

SIAR: Self-Induction Augmented Retrieval, project the query to rule space

Self-Induction Prompt Template

You are given a Query. Please write the inferential rule may help answer the question. The rule should summarize and abstract the facts in the query and catch the underlying logic. I will give some examples. Just output the rule and do not output anything else.
query: $\{q_1\}$
rule: $\{r_1\}$
... more demonstrations



Improve Rule Retrieval and Reasoning with Self-Induction and Relevance ReEstimate - ACL Anthology

Rule Retriever Matters

Rule Relevance ReEstimate: prompt the LLM to directly output a ranked list of rules

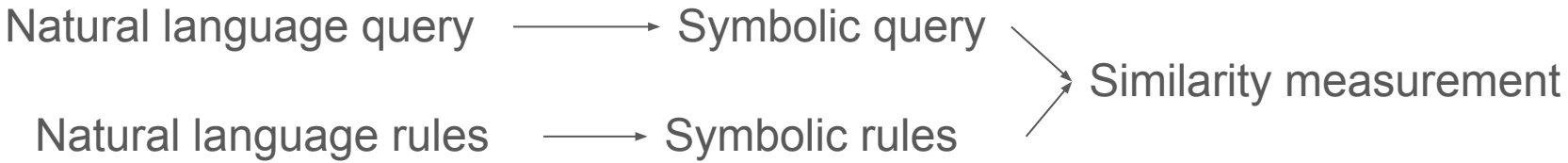
Rule Relevance ReEstimate:

If the abstract knowledge in a rule can be instantiated into the facts in query, that rule is more relevant.
If a rule can be applied to the query and thus be helpful for reasoning, that rule is more relevant.
...
Output the ranking list by the relevance in descending order.
...
query: {q}
[1]. {r1} ... [n]. {rn}

Improve Rule Retrieval and Reasoning with Self-Induction and Relevance ReEstimate - ACL Anthology

Rule Retriever Matters

Map both queries and rules in **symbolic** space using LLM



R: brother_of(C, A) :- sister_of(B, A), brother_of(C, B)
F1: grandson_of(John, James) **F2:** sister_of(Mary, John) predicate unmatched
F2: sister_of(Mary, John) **F3:** brother_of(James, Mary) predicate matched

R: brother_of(C, A) :- sister_of(B, A), brother_of(C, B)
F2: sister_of(Mary, John) **F3:** brother_of(James, Mary) variable matched
F2: sister_of(Mary, John) **F4:** brother_of(Clarence, Timmy) variable unmatched

Figure 4: Examples of predicate and variable matching.

[Symbolic Working Memory Enhances Language Models for Complex Rule Application]

Proposed Rule	If the user query involves identifying a specific familial relationship (e.g., maternal grandfather), then decompose the task by first resolving intermediate relationships (e.g., mother, father) sequentially.
Symbolic Form	`if domain=FAMILIAL_RELATIONSHIP or tool_category=GENEALOGY_QUERY, then action=[DECOMPOSE_QUERY, RESOLVE_INTERMEDIATE_ENTITY, SEQUENCE_SUBTASKS] strength=MANDATORY`

[RIMRULE: Improving Tool-Using Language Agents via MDL-Guided Rule Learning]

Section Summary: Rule-Augmented Generation

- Multiple tasks:
  - reward modeling, text generation, reasoning
- Manage context
  - Memory management of variables and rules
  - Rules retriever
- Apply rules
  - LLM, python or symbolic solver

Industry Applications

Why Industry Cares About Rules

- Reliability over raw performance
- Predictable behavior
- Auditability and compliance

Developer-Facing Rules

- Coding conventions
- Style and formatting rules
- Tool usage constraints

Following User-Specified Rules in AI-Assisted Coding

CURSOR | Search docs... ⌘K | Ask AI ⌘I | Sign in

Core

Rules

Rules provide system-level instructions to Agent. They bundle prompts, scripts, and more together, making it easy to manage and share workflows across your team.

Cursor supports four types of rules:

Project Rules

Stored in `.cursor/rules`, version-controlled and scoped to your codebase.

User Rules

Global to your Cursor environment. Used by Agent (Chat).

Team Rules

Team-wide rules managed from the dashboard. Available on Team and Enterprise plans.

AGENTS.md

Agent instructions in markdown format. Simple alternative to `.cursor/rules`.

Rule Type	Description
`Always Apply`	Apply to every chat session
`Apply Intelligently`	When Agent decides it's relevant based on description
`Apply to Specific Files`	When file matches a specified pattern
`Apply Manually`	When @-mentioned in chat (e.g., `@my-rule`)

```
---
globs:
alwaysApply: false
---

- Use our internal RPC pattern when defining services
- Always use snake_case for service names.

@service-template.ts
```

Cursor supports AGENTS.md in the project root and subdirectories.

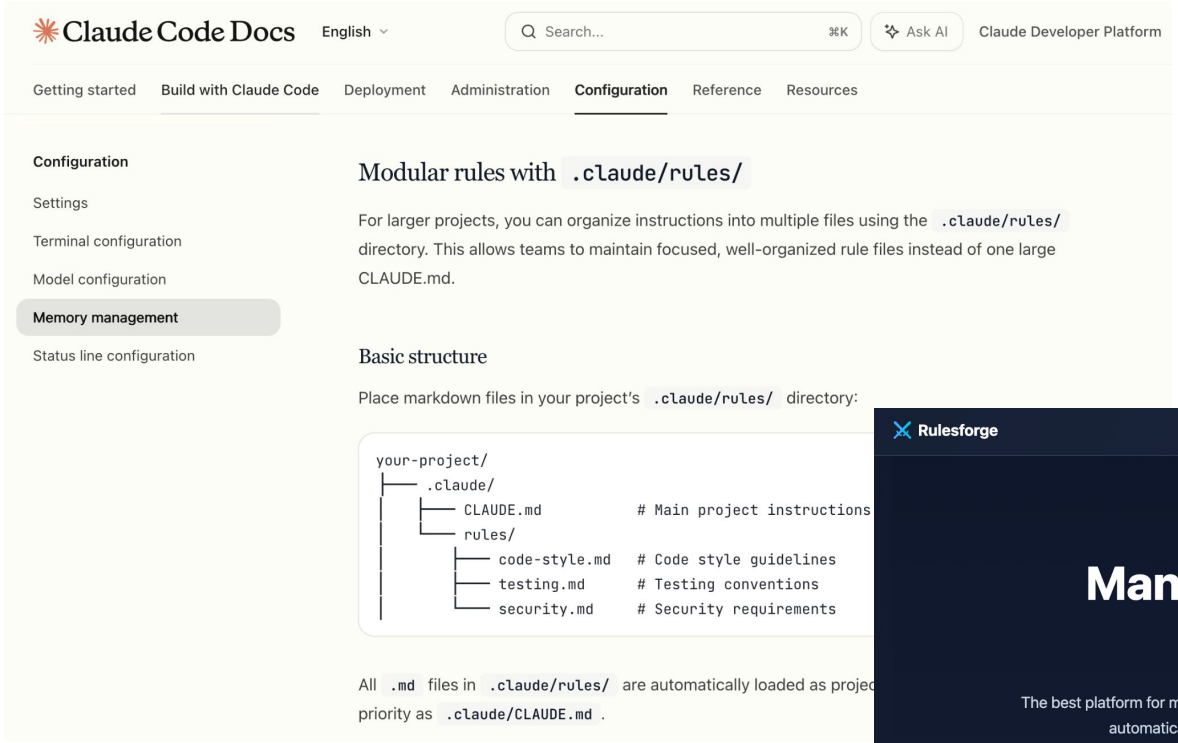
```
# Project Instructions

## Code Style

- Use TypeScript for all new files
- Prefer functional components in React
- Use snake_case for database columns

## Architecture

- Follow the repository pattern
- Keep business logic in service layers
```

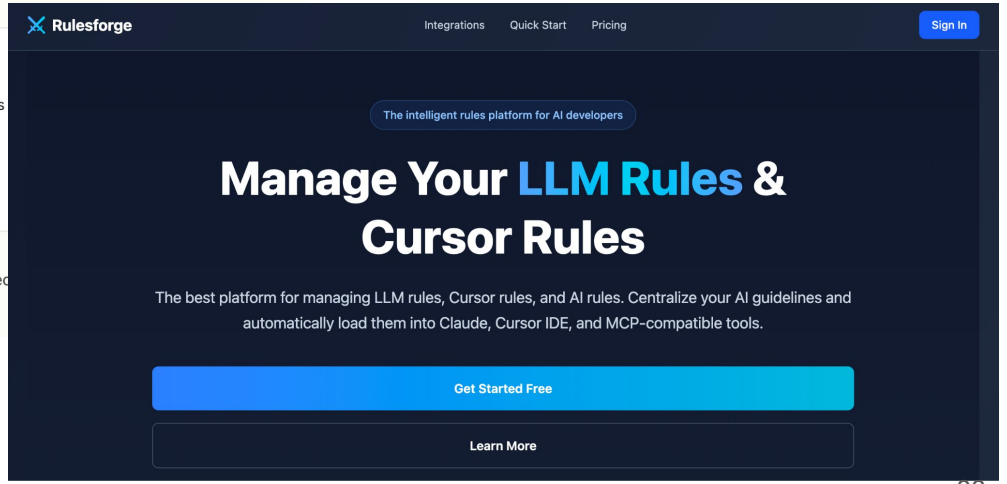
User Rules

User Rules are global preferences defined in **Cursor Settings → Rules** that apply across all projects. They are used by Agent (Chat) and are perfect for setting preferred communication style or coding conventions:

```
Please reply in a concise style. Avoid unnecessary repetition or filler language
```

Following User-Specified Rules in AI-Assisted Coding





https://rulesforge.app/

Helping Users Comply with Complex Rules

BUSINESS INSIDER

Subscribe

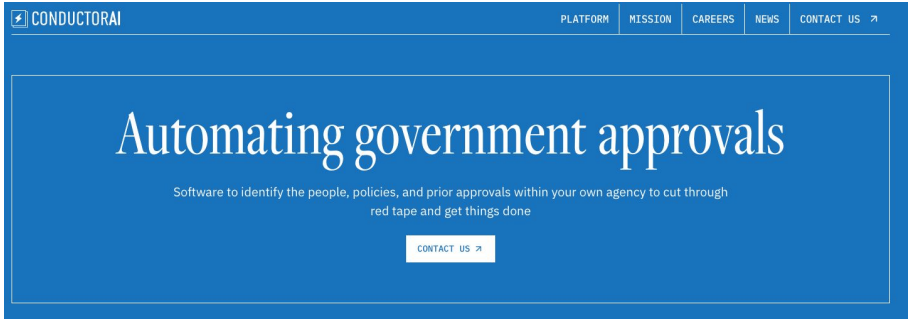
+0.48% NASDAQ +1.02% S&P 500 +0.65% AAPL -0.06% NVDA +0.06% MSFT -0.02% AMZN +0.05% META -0.0

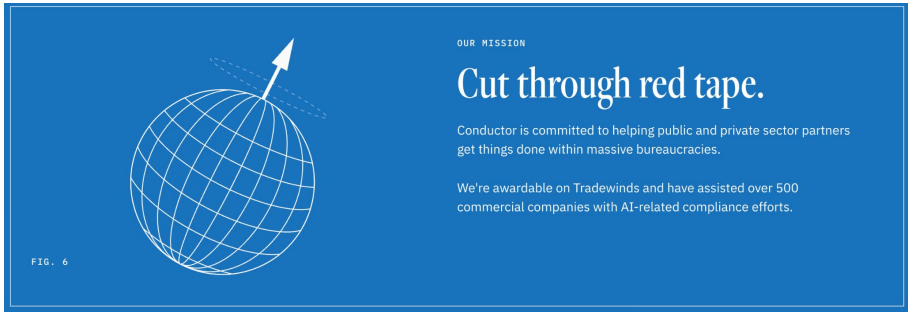
EXCLUSIVE

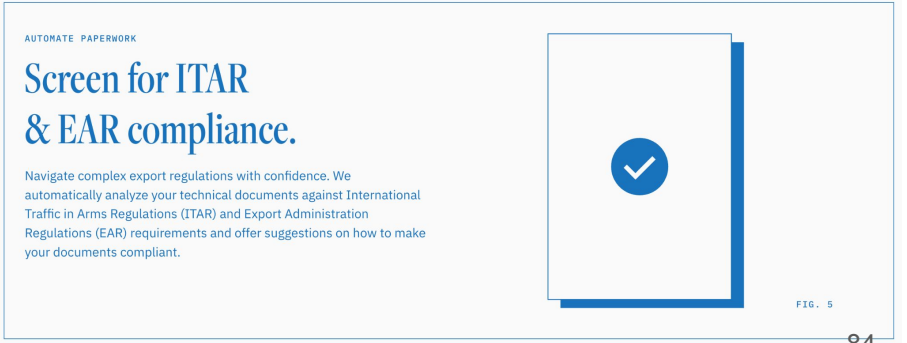
This Palantir alumni-founded startup uses AI to cut through government red tape. It just raised around $15 million, led by Lux Capital.

By Julia Hornstein + Follow

The 15-person company develops an AI platform that can navigate complex government approval workflows in classified environments. Using AI, Conductor's software ingests thousands of pages of complex policy and compliance rules, atomizes them into individual line items, and then determines what is allowable for a given document review or approval process based on the new information in its system.



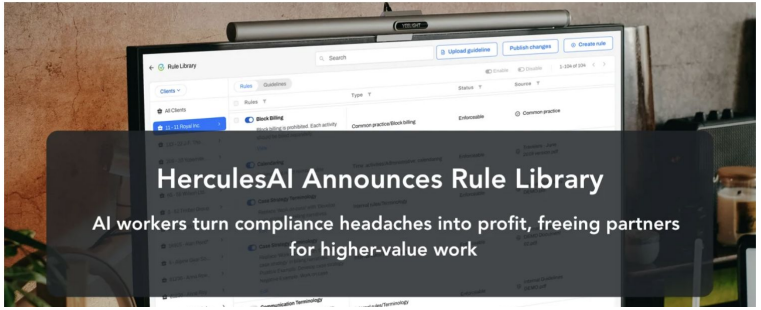


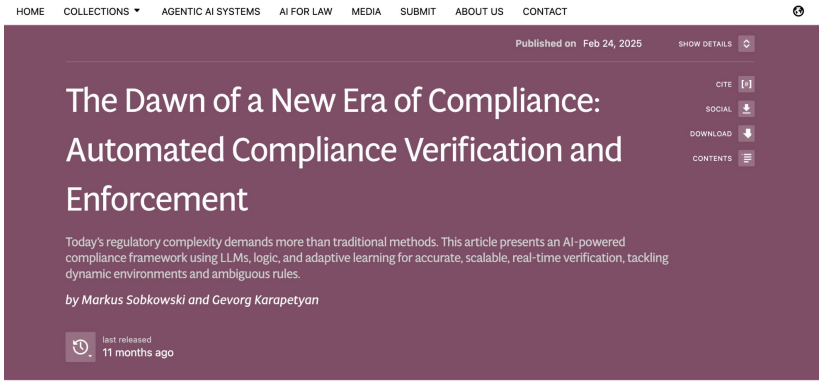


https://www.businessinsider.com/startup-conductor-ai-raises-vc-funding-lux-capital-2025-4

Helping Users Comply with Complex Rules

HerculesAI Rule Library Enforces Law Firm Billing Compliance with Outside Counsel Guidelines





Key capabilities

• **Instant guideline ingestion and extraction** – Drag-and-drop guideline PDFs; the platform's LLM turns prose into structured rules in minutes, citing the source from the document and classifying the rule type.

• **Four rule sources, one library** – Common-practice AI models trained on historical deductions, OCG-derived rules, internal guideline rules covering firmwide time entry standards or specific rules for clients, matters, and partners, as well as ad-hoc custom rules created in natural language live side-by-side for transparent end-to-end governance.

• **Flexibility built into every step** – The billing team can easily enable and disable rules, allowing for exception handling at any level, modification of rules and their attributes, and publishing changes to the Rule Library for immediate enforcement or reverification.

• **Beyond enforcement, suggestions** – The Rule Library, powered by LLMs specifically fine-tuned to carry out legal billing compliance, doesn't just enforce rules, but also provides one-click fix suggestions, cites the source of the rule, and provides transparent AI-reasoning to explain what is causing the non-compliance.

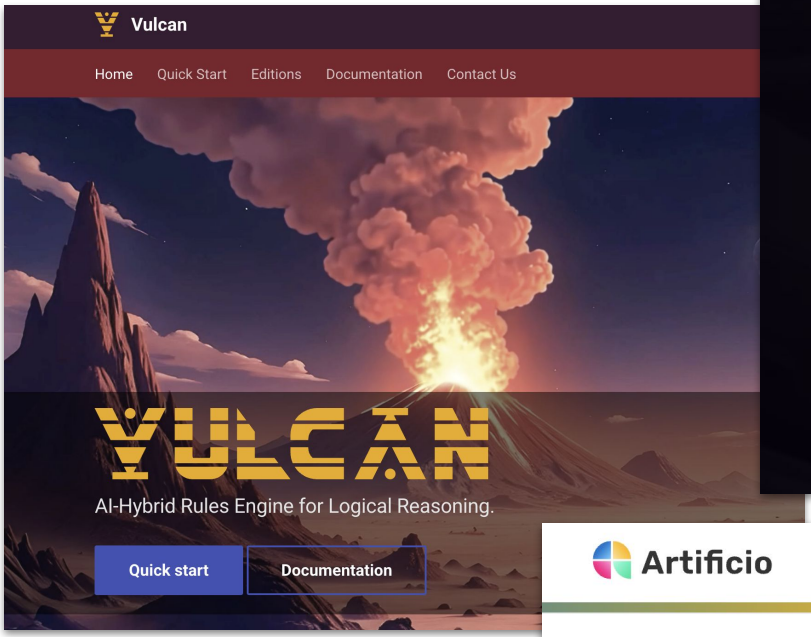
https://www.lawnext.com/2025/05/49882.html

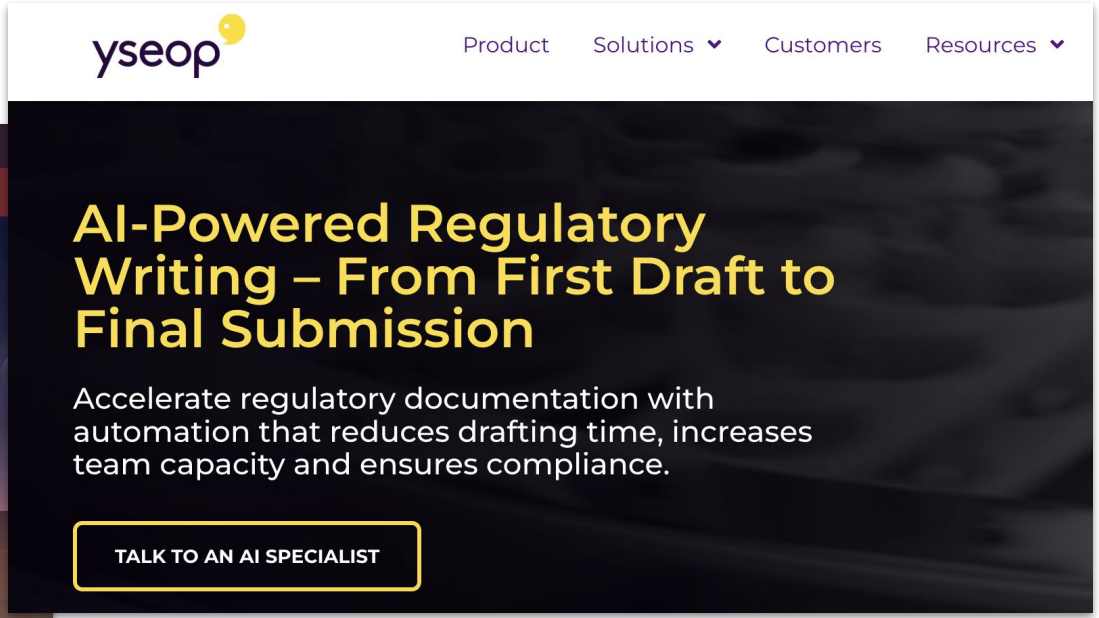
https://law.mit.edu/pub/thedawnofaneweraofcompliance/release/1

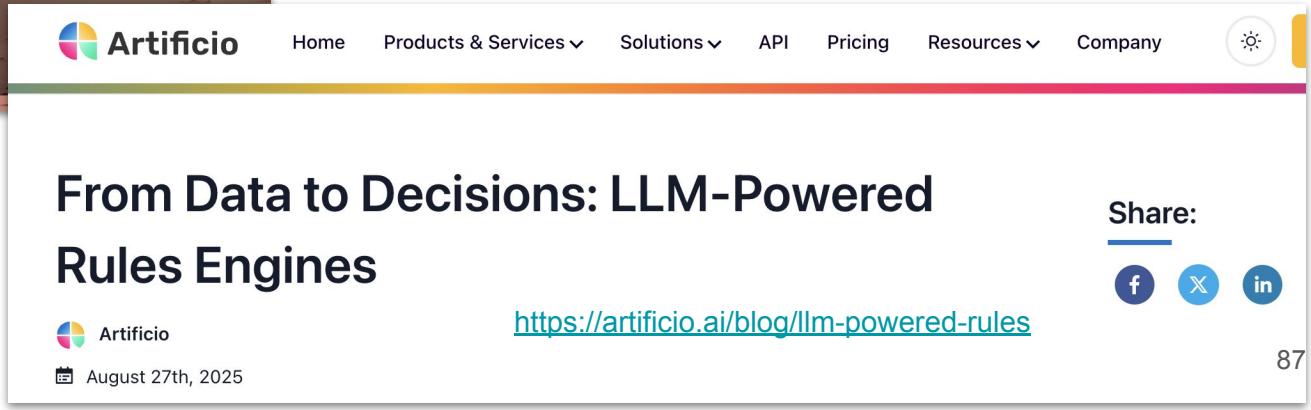
Domain-Constrained Systems

- Healthcare assistants
- Legal research tools
- Customer support agents

And many others…







Lessons from Deployment

- Rules reduce risk, not eliminate it
- Tooling matters
- Human oversight still required

From Practice Back to Research

- Industry needs:
  - Better rule learning
  - Better evaluation
  - Better compositionality
- Research opportunities
  - Richer rule representations
  - Hybrid neural–symbolic training
  - Dynamic rule learning during deployment
  - Rule-aware objectives and decoding

Summary

Key Takeaways

- Rules complement LLMs
- LLMs enable scalable rule discovery
- Rule-augmented generation improves reliability
- Many open research opportunities remain

https://github.com/golsun/rule_learning