

Leveraging Large Language Models for Building Interpretable Rule-Based Data-to-Text Systems

Jędrzej Warczyński, Mateusz Lango, Ondrej Dusek



Problem definition

- **Data-to-text:** generate textual description for given data
- We focus on data expressed as **RDF triples**
- Example:

(Alex Plante, birth year, 1989), (Alex Plante, birth place, Manitoba)

⇒

Alex Plante was born in 1989 in Manitoba.

Two approaches towards data-to-text

- **Rule-based systems**

- manually written by human experts
- interpretable
- controllable
- typically quite fast, run on CPU

- **Neural systems**

- automatically trained by machine learning algorithm
- black-box models
- hallucinations
- need GPU/special hardware to deploy

Two approaches towards data-to-text

- **Rule-based systems**

- manually written by human experts
- interpretable
- controllable
- typically quite fast, run on CPU

- **Neural systems**

- automatically trained by machine learning algorithm
- black-box models
- hallucinations
- need GPU/special hardware to deploy

- **RuLLeM** (this work): neurosymbolic approach using **Large Language Model** to write/train **rule-based** system in pure Python

- automatically trained by machine learning algorithm
- interpretable & controllable
- reduced hallucinations (+ can be fixed manually)
- very fast generation, runs on CPU

RuLeM - overall system structure

We start from a simplistic data2text implementation, containing:

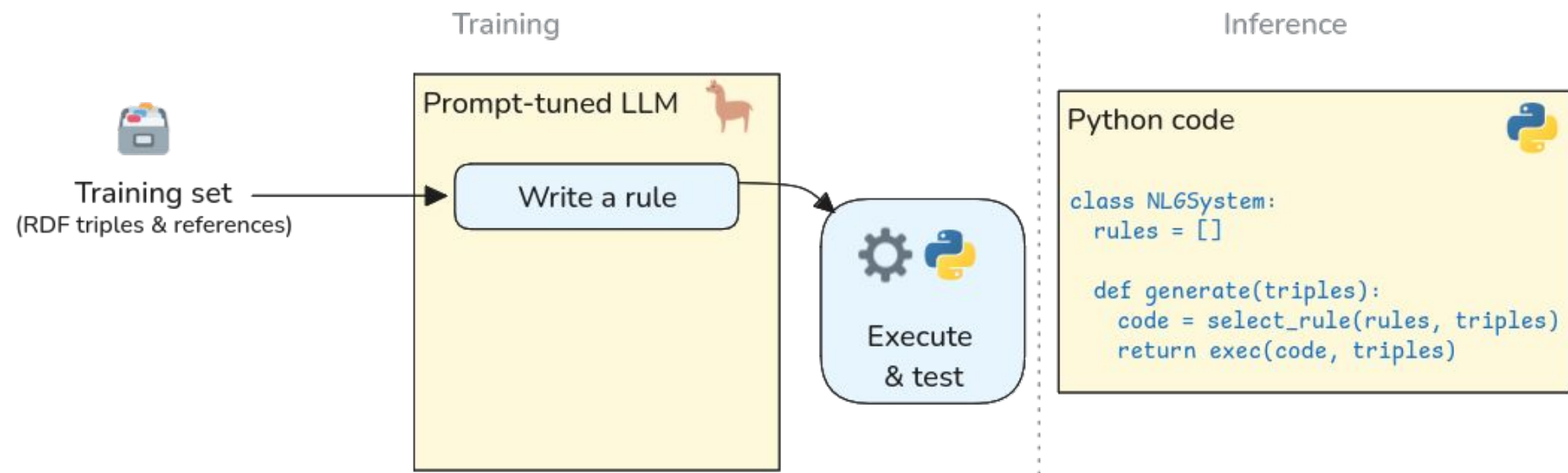
- (empty) structure for storing rules
 - rule *specification*:
 - # input triples,
 - list of their predicates
 - rule code: Python code snippet
- rule selection procedure:
 - rule with matching specification
→ run it
 - else: iteratively find rules to cover subsets
→ run them & concatenate outputs

Python code



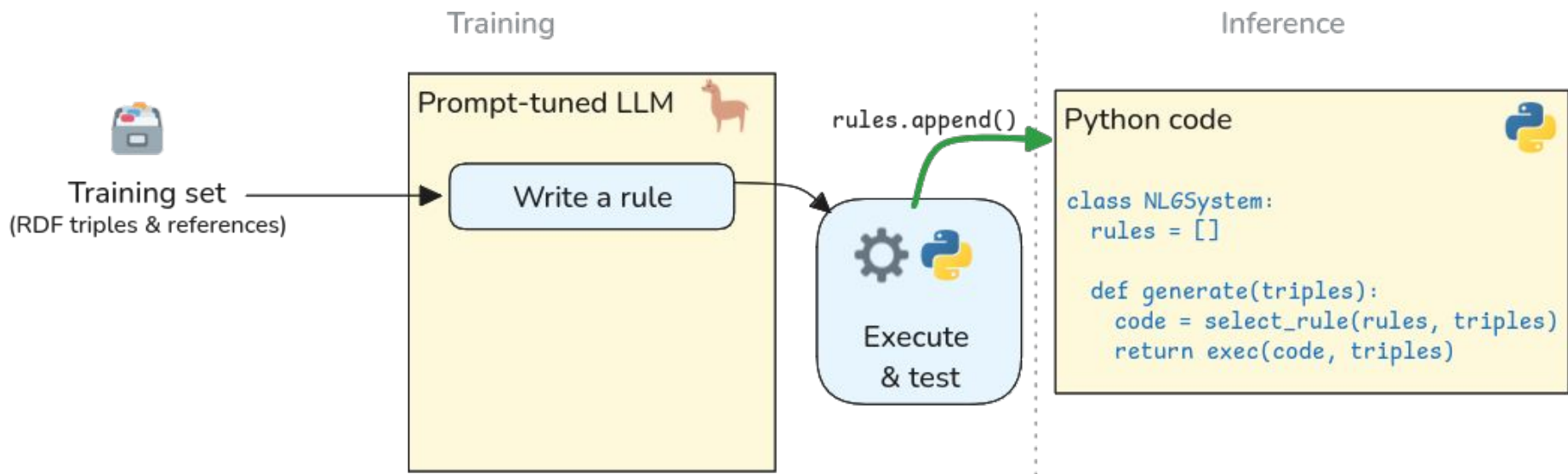
```
class NLGSystem:  
    rules = []  
  
    def generate(triples):  
        code = select_rule(rules, triples)  
        return exec(code, triples)
```

RuLeM - training the system (1/4)



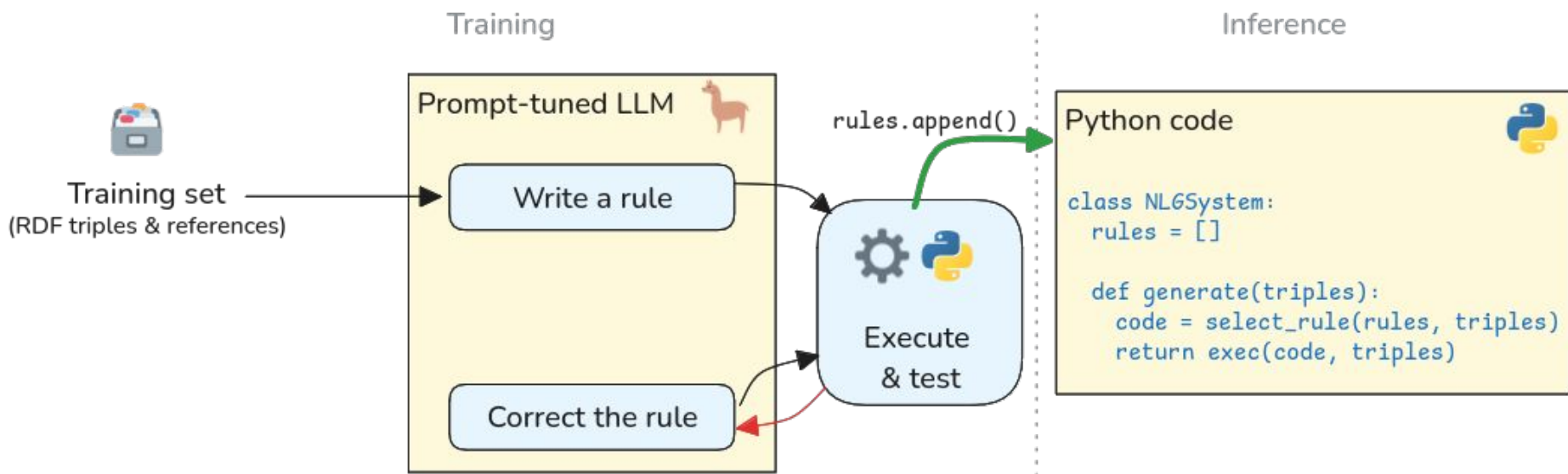
- We iterate over training examples & prompt LLM to write a rule for each
- The prompt contains:
 - Instruction to generate Python code, specifying input & output
 - “make it general to produce outputs for other triples”
 - Code snippet with overall program structure

RuLeM - training the system (2/4)



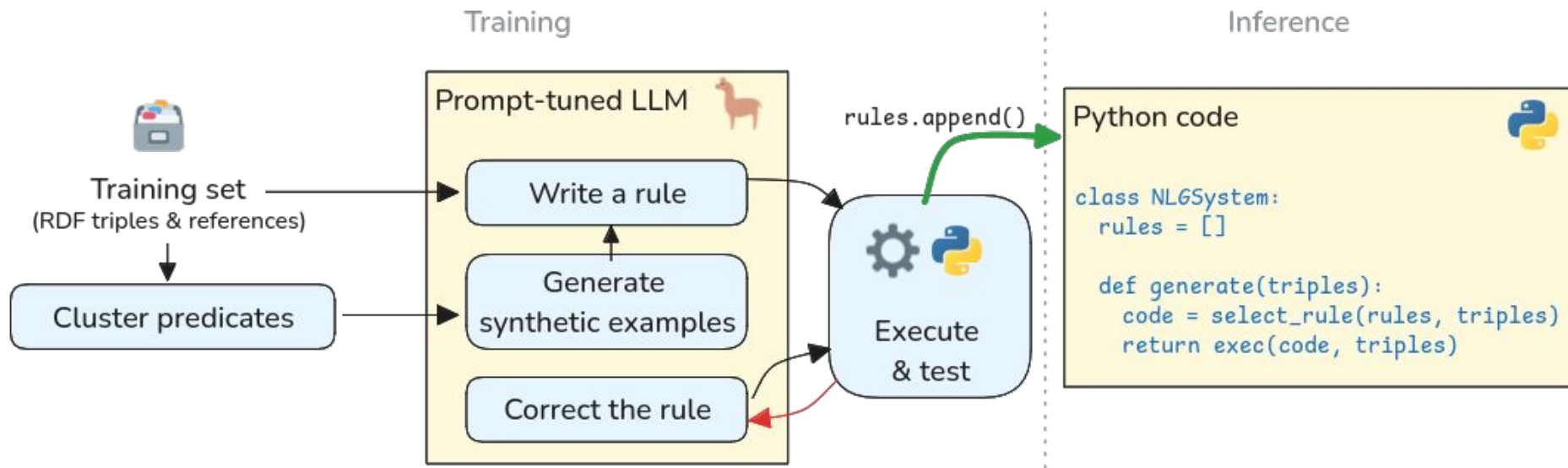
- The rule code is executed and tested.
- We assume the rule is correct if:
 - there's no syntax error
 - execution does not exceed timeout
 - Levenshtein distance between output & reference is within threshold

RuLeM - training the system (3/4)



- Incorrect rule → we ask LLM to fix it, given expected & received output
- Repeated 2x if needed

RuLeM - training the system (4/4)






- To improve generalizability, we construct rules for synthetic examples
 - We cluster triples based on co-occurrence in training set
 - LLM generates synthetic references for all pairs, triples and quadruples of triples within cluster

Experimental setup

- WebNLG dataset
- The performance measured on in-domain data only
- 2 neural baselines
 - Fine-tuned BART model
 - Prompted Llama 3 70B
- RuLLeM uses Llama 3 70B to produce rules
- 3,408 rules constructed from original training set

Automatic evaluation

Method	Automatic metrics			Inference time		Interpretability
	BLEU	METEOR	BLEURT	GPU	CPU	
Llama 3 70B	38.26	<u>0.680</u>	0.113	6,360 s	n/a	
Fine-tuned BART	53.28	0.716	0.257	249 s	1,910 s	
RuLLeM	<u>42.51</u>	0.671	<u>0.157</u>	-	3s	

Human evaluation

- 75 instances from the test set of WebNLG annotated by 5 NLP experts for 3 systems (i.e. 225 system outputs in total)
- 5 binary questions, order of outputs randomized

	minor hallucinations	major hallucinations	omissions	disfluencies	repetitions
Llama 3 70B	<u>0.08</u>	0.07	0.07	0.19	0.03
Fine-tuned BART	0.20	0.33	0.19	<u>0.16</u>	0.07
RuLLeM	0.04	<u>0.13</u>	<u>0.08</u>	0.13	0.03

Other experiments

What about other LLMs than Llama 3 70B?

- We checked smaller LLMs: Llama 3 7B, Mistral 7B, Codellama 7B
- Not satisfactory, **large model seems to be needed** to develop rules

Is the code of RuLLeM truly interpretable for a human?

- We chose 5 hallucinations found in human evaluation and asked a Python programmer to fix it (without AI tools)
- The programmer got familiar with the code & **successfully fixed all the errors** in <15 minutes (3 min./example).

Summary

- We present an idea of **using LLM to implement a NLG system**
- The framework is very simple
(e.g. no LLM fine-tuning, no optimization of NLG metrics)
- But the output quality is
~ between few-shot prompted LLM and finetuned BART
- RuLLeM offers **full interpretability** and **very fast generation**

Thanks

Email: lango@ufal.mff.cuni.cz

Paper: <https://aclanthology.org/2024.inlg-main.48.pdf>

Code: <https://github.com/jwarczynski/RuLLeM>

*This research was funded by the European Union (ERC, NG-NLG, 101039303)
and National Science Centre, Poland (Grant No. 2022/47/D/ST6/01770).
It used resources of the LINDAT/CLARIAH-CZ Research Infrastructure (Czech MEYS LM2018101).*



Leveraging Large Language Models for Building Interpretable Rule-Based Data-to-Text Systems

Jędrzej Warczyński, Mateusz Lango, Ondrej Dusek

