Correct and Optimal: the Regular Expression Inference Challenge

M. Valizadeh[1, 2] P. J. Gorinski[3, 4] I. Iacobacci[4] M. Berger[1, 5]

[1] University of Sussex

[2] Neubla UK Ltd

[3] Robin AI

[4] Huawei Noah's Ark Lab, London

[5] Montanarius Ltd

Jeju, 9 August 2024

We introduce new benchmark for code learning

Background

DNNs, particularly transformers have captured the attention of the world.

Applications in logic and verification, where 100% precision is required, are dominated by symbolic methods.

This raises the question of comparison.

Good, apples-to-apples benchmarks are in short supply.

Background

Requirements:

- Needs lots of training data for transformers
- Symbolic AI approaches typically don't handle large amount of training data
- Also: problem domain should be natural, general, and easy to quantify
- Also: tunable hardness from trivial to way beyond the SOTA

INFORMATION AND CONTROL **10,** 447–474 (1967)

Language Identification in the Limit

E MARK GOLD*

The RAND Corporation

Language learnability has been investigated. This refers to the following situation: A class of possible languages is specified, together with a method of presenting information to the learner about an unknown language, which is to be chosen from the class. The question

Regular expression inference (REI)

Given two sets P and N of strings:

P: 10, 101, 100, 1010, 1011, 1000, 1001

N: ϵ, 0, 1, 00, 11, 010

Learn the regular expression that accepts **all** strings in P and rejects **all** strings in N!

Regular expression inference (REI)

Given two sets P and N of strings:

P: 10, 101, 100, 1010, 1011, 1000, 1001

N: ϵ, 0, 1, 00, 11, 010

Learn the regular expression that accepts **all** strings in P and rejects **all** strings in N!

Trivial solution (overfitting on P):

$$10 + 101 + 100 + 1010 + 1011 + 1000 + 1001$$

Regular expression inference (REI)

Given two sets P and N of strings:

P: 10, 101, 100, 1010, 1011, 1000, 1001

N: ϵ, 0, 1, 00, 11, 010

Learn the **smallest** regular expression that accepts **all** strings in P and rejects **all** strings in N!

Trivial solution (overfitting on P):

$$10 + 101 + 100 + 1010 + 1011 + 1000 + 1001$$

Regular expressions (REs)

REs: a canonical and minimal model of a programming language.

Syntax where Σ is alphabet:

$$
r, r' \quad ::= \quad \begin{cases} \varnothing & \text{The empty set} \\ \epsilon & \text{The empty string} \\ a & \text{Alphabet character } a \in \Sigma \\ r^* & \text{0 or more repetitions of } r \\ r \cdot r' & \text{Concatenation} \\ r + r' & \text{Union, logical disjunction} \\ \ldots & \end{cases}
$$

Regular expressions: cost homomorphism

A **cost homomorphism** is a map $\text{cost}(\cdot) : \text{RE}(\Sigma) \to \mathbb{N}$ s. t.

- $\text{cost}(\varnothing) = c_1$
- $\text{cost}(r^*) = \text{cost}(r) + c_3$
- $\text{cost}(r \cdot r') = \text{cost}(r) + \text{cost}(r') + c_5$
- $\text{cost}(r + r') = \text{cost}(r) + \text{cost}(r') + c_7$
- ...

for constants $c_1, \ldots, c_8 > 0$.

Note: cost homomorphism is given by an 8-tuple, so **easy to learn**.

Example: intuitive RE size is $c_1 = \cdots = c_8 = 1$.

Regular expression inference (REI)

- **Input:** Two finite sets of strings P and N of positive and negative examples, and a cost homomorphism $\text{cost}(\cdot)$ for REs (given by an 8-tuple).
- **Output:** A regular expression r that is both:
  - **Correct:** Meaning that r accepts all strings in P and rejects all strings in N.
  - **Optimal:** No regular expression with a cost less than $\text{cost}(r)$ is correct.

We do **not** allow strings to be mis-classified.

Performance of learner is rigorously quantified by Δ of cost of learned RE over possible minimum

Regular expression inference (REI)

- **Input:** Two finite sets of strings P and N of positive and negative examples, and a cost homomorphism $\text{cost}(\cdot)$ for REs (given by an 8-tuple).
- **Output:** A regular expression r that is both:
  - **Correct:** Meaning that r accepts all strings in P and rejects all strings in N.
  - **Optimal:** No regular expression with a cost less than $\text{cost}(r)$ is correct.

We do **not** allow strings to be mis-classified.

Performance of learner is rigorously quantified by Δ of cost of learned RE over possible minimum

Make a quantitative prediction in your mind: how will LLMs perform?

Dataset generation problem

How to get unlimited amounts of training data?

How to get oracle (minimal RE for each PN set) that works on large training sets?

Program synthesis on GPUs

The data set generation became possible with a recent (2023/4) work in enumerative program synthesis, when REI was ported to a GPU.

Dataset generation

Random strings with simple model hyperparameters:

- Alphabet size
- Number of strings in PN
- Maximal string length K
- Distribution of length of strings in PN
  - Uniform over all strings of length $< K$
  - Uniform choice of length $0 \leq i < K$, and uniform among strings of length i.

Compute oracle (minimal solution) using program synthesis on GPU!

Test and training data are in-distribution

Competition

REI - Regular Expression Inference

Organized by REIC - Current server time: Aug. 6, 2024, 4:40 a.m. UTC

Previous	▶ Current	End
DS4	Combined	Competition Ends
Aug. 14, 2023, midnight UTC	Aug. 14, 2023, midnight UTC	Never

Welcome to REIC: The Regular Expression Inference Challenge!

What?

Regular Expression Inference (REI) is a supervised machine learning and program synthesis problem. The task aims at finding **minimal**

Neural baselines

StarChatβ instruction-tuned variant of StarCoderPlus (16B parameters), trained on English & 80+ programming languages. Performs REI with a few-shot prompt.

ReGPT is a GPT-2-like model (300M parameters) trained on the REIC training data (252,947 PN sets).

Measurements of baselines

Test data: 23,489 PN sets

	Correct syntax	Prec%	P%	N%	PN%	Min	Min% P	**Min% G**	Cost Ratio
StarChatβ	99.86	0.4	42.1	79.5	63.2	5	5.4	<0.1	2.16
ReGPT-1k	100	85.9	97.2	97.6	97.7	3180	15.8	13.5	1.09
ReGPT-10	99.99	82.6	95.3	95.8	96.0	2924	15.1	12.4	1.09

StarChatβ is prompted to generate 10 REs for each test instance.

ReGPT-1k and ReGPT-10 are ReGPT model with 1k and 10 sampled solutions, respectively.

We keep the best solution as measured by PN Ratio;

Tangent on noisy REI

Experiment on the complexity of noisy REI (n = 1):

Allowed Error	# REs	Minimal RE	Size(RE)
0 %	26,774,099,142	10?+0?(00+10*10?(0+1))1?	28
5 %	319,649,322	((0+1)0+(0+11)*1)(100?)?	22
10 %	18,698,767	(10+0*1)(10?(0+1))?	18
15 %	794,598	(0+1)0+(0+11)*1	14
20 %	116,912	(0+11)*(1+00)	12
25 %	2,073	(0+11)*1	8
30 %	2,073	(0+11)*1	8
35 %	1,124	1+(0+1)0	7
40 %	50	10?	4
45 %	3	1	1
50 %	1	$\emptyset$	1

Conjecture (noisy REI complexity)

There is an exponential dependency between the allowed error ϵ, and the hardness of REI with allowed error ϵ.

Discussion of Baseline Performance

ReGPT finds cost-efficient solutions, but rarely optimal ones. Sampling 10 or 1000 REs makes little difference.

Reasonable performance of ReGPT suggests hybrids: a fine-tuned DNN providing solutions close to minimal as starting points for symbolic methods.

Prompted StarChatβ performs badly. While it is large and trained for code synthesis, a superset of REI, it has never seen REIC data during training. So we may wonder about its ability to generalise.

Thank you

Challenge

Paper

https://codalab.lisn.upsaclay.fr/competitions/15096
https://arxiv.org/abs/2308.07899

Energy consumption

This benchmark would be much more interesting if we could relate the benchmark performance with compute used (measured by e.g., energy consumption).

Doing this well is an open problem!