What Makes Instruction Learning Hard? An Investigation and a New Challenge in a Synthetic Environment

Matthew Finlayson Kyle Richardson Ashish Sabharwal Peter Clark




matthewf@allenai.org

EMNLP 2022

mattf1n.github.io

TL;DR

What Makes Instruction Learning Hard?
An Investigation and a New Challenge in a Synthetic Environment

Matthew Finlayson **Kyle Richardson** **Ashish Sabharwal** **Peter Clark**

Allen Institute for AI, Seattle, WA
{matthewf,kyler,ashishs,peterc}@allenai.org

TL;DR

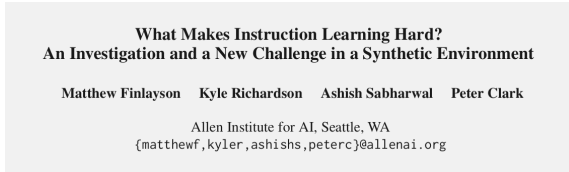
What Makes Instruction Learning Hard?
An Investigation and a New Challenge in a Synthetic Environment

Matthew Finlayson **Kyle Richardson** **Ashish Sabharwal** **Peter Clark**

Allen Institute for AI, Seattle, WA
{matthewf,kyler,ashishs,peterc}@allenai.org

- Motivation: instruction learning

TL;DR



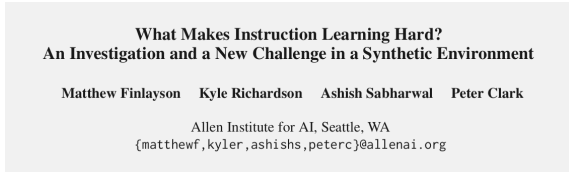
What Makes Instruction Learning Hard?
An Investigation and a New Challenge in a Synthetic Environment

Matthew Finlayson **Kyle Richardson** **Ashish Sabharwal** **Peter Clark**

Allen Institute for AI, Seattle, WA
{matthewf,kyler,ashishs,peterc}@allenai.org

- Motivation: instruction learning
- What types of instructions can LMs follow?

TL;DR



What Makes Instruction Learning Hard?
An Investigation and a New Challenge in a Synthetic Environment

Matthew Finlayson **Kyle Richardson** **Ashish Sabharwal** **Peter Clark**

Allen Institute for AI, Seattle, WA
{matthewf,kyler,ashishs,peterc}@allenai.org

- Motivation: instruction learning
- What types of instructions can LMs follow?
- Regular languages as instructions

Regular languages and expressions

Regular languages and expressions

- A *formal language* is a set of strings over some alphabet.

E.g., the set of strings containing "a" ($\{a, aa, ab, \ldots\}$) is a language over the alphabet $\{a, b\}$.

Regular languages and expressions

- A *formal language* is a set of strings over some alphabet.
- The *regular languages* are a family of formal languages.

E.g., the set of strings with an even number of “a”s is regular. The set of strings with matching parentheses is not.

Regular languages and expressions

- A *formal language* is a set of strings over some alphabet.
- The *regular languages* are a family of formal languages.
- *Regular expressions* (RegExs) describe regular languages succinctly.

E.g., RegEx $\mathrm{a}|\mathrm{ba}$ ("a or ba") expresses the language $\{\mathsf{a}, \mathsf{ba}\}$.
RegEx $\mathrm{a}*$ expresses $\{\mathsf{a}, \mathsf{aa}, \mathsf{aaa}, \ldots\}$.

Instruction learning

Instruction learning

Instruction learning

$$F : \mathcal{R} \to \mathcal{X} \to \mathcal{Y}$$

- Instruction language $\mathcal{R}$
- Input space $\mathcal{X}$
- Output space $\mathcal{Y}$

Instruction learning

$$F : \mathcal{R} \rightarrow \mathcal{X} \rightarrow \mathcal{Y}$$

- Instruction language $\mathcal{R}$ (e.g., English)
- Input space $\mathcal{X}$ (e.g., movie reviews)
- Output space $\mathcal{Y}$ (e.g., positive/negative)

RegEx as instruction learning

Regular languages have *well-studied properties* that we can use to characterize transformer instruction learning abilities. 🔍

RegEx as instruction learning

Regular languages have *well-studied properties* that we can use to characterize transformer instruction learning abilities. 🔍

$$F : \mathcal{R} \to \mathcal{X} \to \mathcal{Y}$$

- $\mathcal{R} =$ RegExs

RegEx as instruction learning

Regular languages have *well-studied properties* that we can use to characterize transformer instruction learning abilities. 🔍

$$F : \mathcal{R} \to \mathcal{X} \to \mathcal{Y}$$

- $\mathcal{R} =$ RegExs
- $\mathcal{X} = \Sigma^*$
- $\mathcal{Y} = \{\top, \bot\}$

RegSet

Train			Test		
Instruction (RegEx)	Input (String)	Output (T/F)	Instruction (RegEx)	Input (String)	Output (T/F)
$a*b$	aaa	F	$(a*b)*$	aabab	T
$a*b$	aab	T	$(a*b)*$	aba	F
$(ab)*$	aab	F	$(a*b)*$	aab	T
$(ab)*$	abab	T	$a*$	aab	F
$(ab)*$	aabab	F	$a*$	aaa	T
...	...	...	...	...	...

Sampling criteria

Sampling criteria

- Uniform w.r.t. RegEx size

Sampling criteria

- Uniform w.r.t. RegEx size
- Uniform w.r.t. input size

Sampling criteria

- Uniform w.r.t. RegEx size
- Uniform w.r.t. input size
- Balanced examples per RegEx

Sampling criteria

- Uniform w.r.t. RegEx size
- Uniform w.r.t. input size
- Balanced examples per RegEx
- Languages represented uniquely and concisely e.g., $\mathrm{a}|\mathrm{b} = \mathrm{a}|\mathrm{b}|\mathrm{a}|\mathrm{b}$.

Training and evaluation

Training and evaluation

- ByT5-Large to avoid tokenization issues.

Training and evaluation

- ByT5-Large to avoid tokenization issues.
- Evaluation: accuracy is misleading.

Training and evaluation

- ByT5-Large to avoid tokenization issues.
- Evaluation: accuracy is misleading.
- We measure the proportion of RegExs for which accuracy is over 90%.

$$\text{perf@90} = \overbrace{\mathbb{E}_{r \in D} \mathbb{1}(0.9 < \underbrace{\mathbb{E}_{(x,y) \in D_r} \mathbb{1}(M(r,x) = y)}_{\text{accuracy on } r})}^{\text{proportion with over 90\% accuracy}}$$

Language-level attributes: starfree

Language-level attributes: starfree

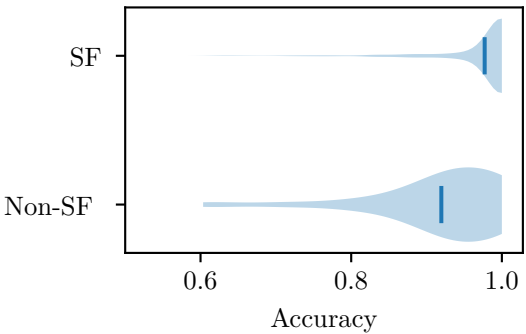
$$\mathrm{a}* = (\emptyset^{\mathrm{c}}\mathrm{b}\emptyset^{\mathrm{c}})^{\mathrm{c}} \in \mathit{SF}$$

Language-level attributes: starfree

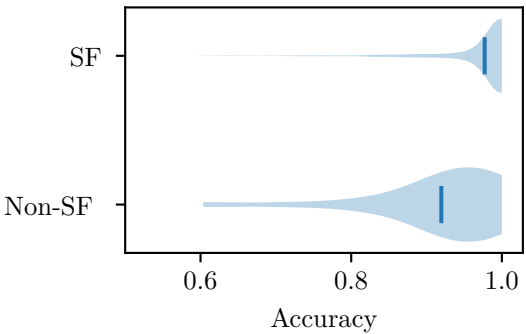
$$\mathrm{a}* = (\emptyset^{\mathrm{c}}\mathrm{b}\emptyset^{\mathrm{c}})^{\mathrm{c}} \in \mathit{SF}$$

$$(\mathrm{aa})* \notin \mathit{SF}$$

Language-level attributes: starfree



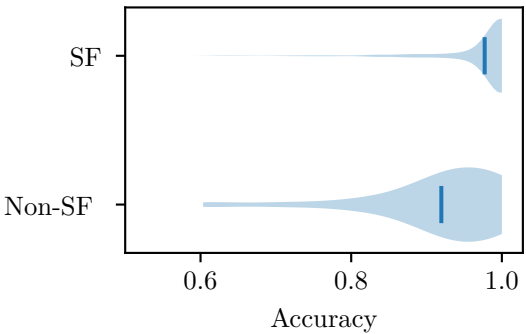
Language-level attributes: starfree



	Starfree	Non-Starfree
perf@90	91.9	71.9

- Non-starfree languages are harder.

Language-level attributes: starfree



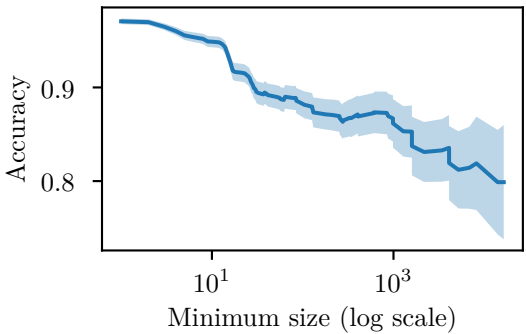
	Starfree	Non-Starfree
perf@90	91.9	71.9

- Non-starfree languages are harder.

Hypothesis: Instructions that require modular counting are hard

Language-level attributes: language size

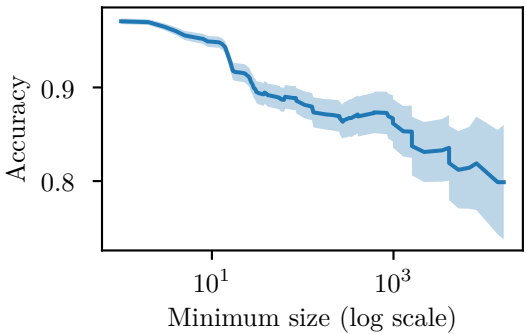
Language-level attributes: language size



	Small (< 64)	Large (≥ 64)
perf@90	95.1	57.5

Bigger languages are harder.

Language-level attributes: language size



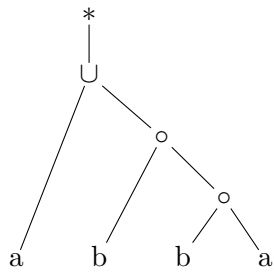
	Small (< 64)	Large (≥ 64)
perf@90	95.1	57.5

Bigger languages are harder.

Hypothesis: abstract instructions are harder

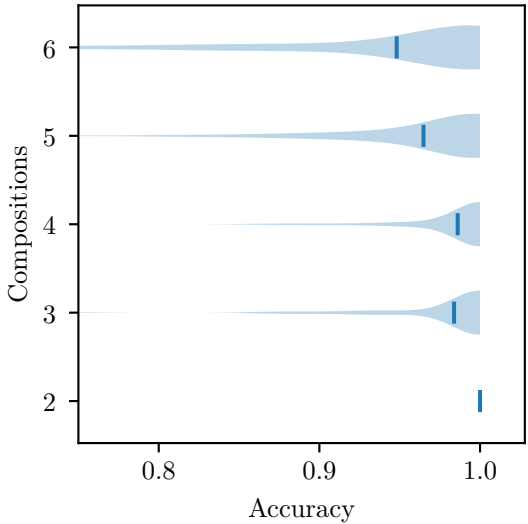
Expression-level attributes: composition

Expression-level attributes: composition



$(\mathrm{a}|\mathrm{bba})*$

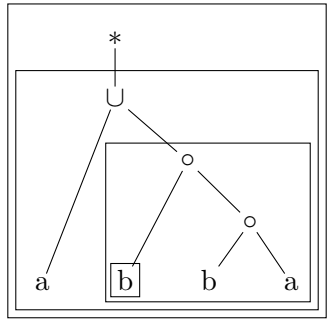
Expression-level attributes: composition



	perf@90
Low (< 6)	95.2
High (≥ 6)	80.3

More composed expressions are somewhat harder.

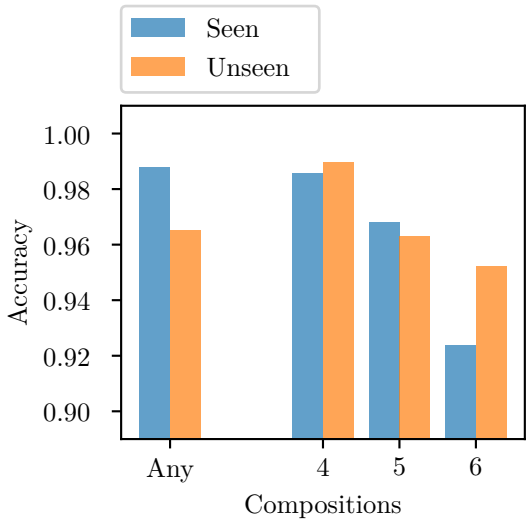
Expression-level attributes: unobserved local structures



$$(\mathrm{a}|\mathrm{bba})*$$

$$\{(\mathrm{a}|\mathrm{bba})*, \mathrm{a}|\mathrm{bba}, \mathrm{a}, \mathrm{bba}, \mathrm{bb}, \mathrm{ba}, \mathrm{b}\}$$

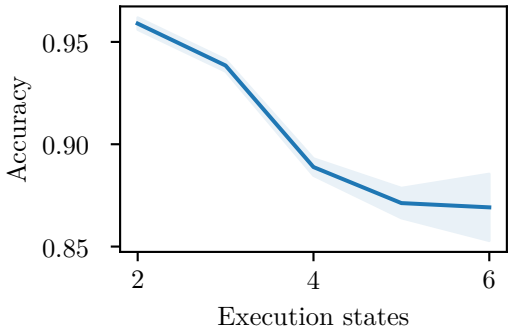
Expression-level: unobserved local structures



Unobserved local structures
do not contribute to hardness.

Instance-level: execution states

The execuation states of a RegEx r and string x is the number of unique states in the minimal DFA for r that are visited while recognizing x.



Instances requiring many execution states are hard.

Hypothesis: instructions that require tracking long contexts are hard.

Hard RegSet

Filter for

- Non-starfree
- Size > 64
- Execution states > 4

Results

Train	Eval	acc	perf@80	**perf@90**	perf@100

Results

Train	Eval		acc	perf@80	**perf@90**	perf@100
Expl.	Expl.	IID	97.1	96.4	**89.6**	69.9
Hard	Hard	IID	88.9	81.6	**65.6**	15.2

Results

Train	Eval		acc	perf@80	**perf@90**	perf@100
Expl.	Expl.	IID	97.1	96.4	**89.6**	69.9
Hard	Hard	IID	88.9	81.6	**65.6**	15.2
Expl.	Hard	OOD	77.2	52.8	**23.4**	2.0
Hard	Expl.	OOD	66.8	29.3	**11.0**	3.8

Hard RegSet remains hard, even in the IID setting.

Summary

Summary

- RegExs are an useful proxy for instruction learning

Summary

- RegExs are an useful proxy for instruction learning
- Models are not good at modular counting, ambiguous instructions, and context-dependent tasks.

Summary

- RegExs are an useful proxy for instruction learning
- Models are not good at modular counting, ambiguous instructions, and context-dependent tasks.
- There are other factors that make RegExs hard for transformers that are yet unknown.