Neural-Symbolic Recursive Machine for Systematic Generalization

Qing Li, Yixin Zhu, Yitao Liang, Ying Nian Wu, Song-Chun Zhu, Siyuan Huang

北京通用人工智能研究院
Beijing Institute for General Artificial Intelligence

Project

ICLR
Vienna Austria, May 7-11, 2024

Introduction

Motivation: how can we build a model with human-like systematic generalization?

Hypothesis: a model with human-like systematic generalization is *compositional* and *equivariant*.

walk right ⟶ RTURN WALK   jump → JUMP

compositional ↓ $\Phi(T_c(x_1, x_2)) = T_c'(\Phi(x_1), \Phi(x_2))$

walk right after jump ⟶ JUMP RTURN **WALK**

equivariant ↓ $\Phi(T_p x) = T_p' \Phi(x)$

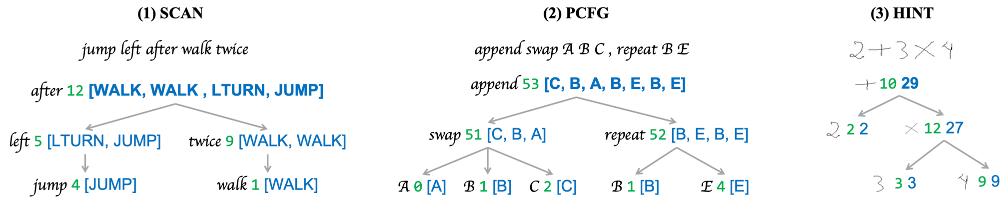
run right after jump ⟶ JUMP RTURN **RUN**

Our contributions:

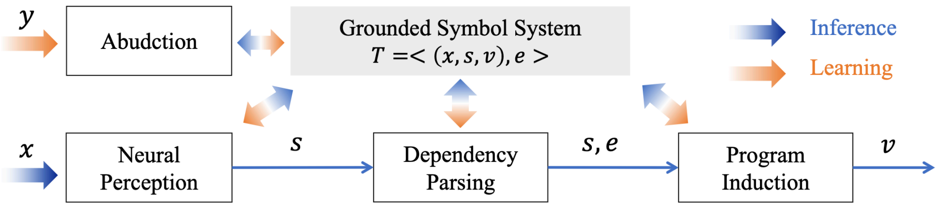
1. We propose Neural-Symbolic Recursive Machine (NSR), a principled framework for jointly learning perception, syntax, and semantics.
2. The core of NSR is a Grounded Symbol System (GSS), allowing for the emergence of combinatorial syntax and semantics directly from training data.
3. NSR achieves impressive systematic generalization across SCAN, PCFG, and HINT.

Code: https://liqing-ustc.github.io/NSR

Grounded Symbol System (GSS)



Neural-Symbolic Recursive Model (NSR)



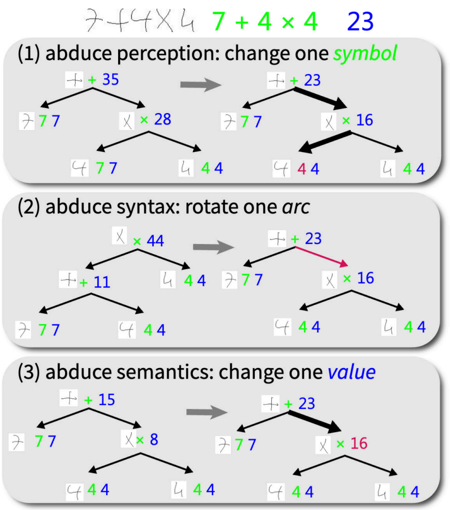
$$p(s|x;\theta_p) = \prod_i p(w_i|x_i;\theta_p) \quad p(e|s;\theta_s) = p(\mathcal{T}|s;\theta_s) = \prod_{t_i \in \mathcal{T}} p(t_i|c_i;\theta_s) \quad p(v|e,s;\theta_l) = \prod_i p(v_i|s_i, \text{children}(s_i);\theta_l))$$

Learning by Deduction-Abduction

Algorithm A1: Learning by Deduction-Abduction

Input : Training set $D = (x_i, y_i) : i = 1, 2, ..., N$
Output : $\theta_p^{(T)}, \theta_s^{(T)}, \theta_l^{(T)}$
Initial Module: perception $\theta_p^{(0)}$, syntax $\theta_s^{(0)}$, semantics $\theta_l^{(0)}$
for $t \leftarrow 0$ *to* T **do**
  Buffer $\mathcal{B} \leftarrow \varnothing$
  foreach $(x, y) \in D$ **do**
    $T \leftarrow \textsc{Deduce}(x, \theta_p^{(t)}, \theta_s^{(t)}, \theta_l^{(t)})$
    $T^* \leftarrow \textsc{Abduce}(T, y)$
    $\mathcal{B} \leftarrow \mathcal{B} \cup T^*$
  $\theta_p^{(t+1)}, \theta_s^{(t+1)}, \theta_l^{(t+1)} \leftarrow \text{learn}(\mathcal{B}, \theta_p^{(t)}, \theta_s^{(t)}, \theta_l^{(t)})$
return $\theta_p^{(T)}, \theta_s^{(T)}, \theta_l^{(T)}$

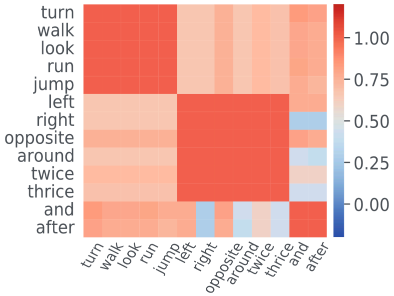
Function $\textsc{Deduce}\ (x, \theta_p, \theta_s, \theta_l)$:
  Sample $\hat{s} \sim p(s|x;\theta_p), \hat{e} \sim p(e|\hat{s};\theta_s), \hat{v} = f(\hat{s}, \hat{e}; \theta_l)$
  return $T =< (x, \hat{s}, \hat{v}), \hat{e} >$



Experimental Results

NSR achieves perfect generalization in SCAN and PCFG.

models	SCAN				PCFG		
	Simple	Jump	Around Right	Length	i.i.d.	systematicity	productivity
Seq2Seq	99.7	1.2	2.5	13.8	79	53	30
CNN	100.0	69.2	56.7	0.0	85	56	31
Transformer	-	-	-	20.0	-	96	85
NeSS	100.0	100.0	100.0	100.0	≈0	≈0	≈0
NSR (ours)	**100.0**	**100.0**	**100.0**	**100.0**	**100**	**100**	**100**



(a) Syntactic similarity amongst input words in NSR.

turn : ∅ → [ ]
walk : ∅ → [inc 0]
look : ∅ → [inc inc 0]
run : ∅ → [inc inc inc 0]
jump : ∅ → [inc inc inc inc 0]
left : x → cons (inc inc inc inc 0, x)
right : x → cons (inc inc inc inc inc 0, x)
opposite : x → cons(car(x), x)
around : x → +(+(+(x, x), x), x)
twice : x → +(x, x)
thrice : x → +(+(x, x), x)
and : x, y → +(x, y)
after : x, y → +(y, x)

1 → WALK
2 → LOOK
3 → RUN
4 → JUMP
5 → LTRUN
6 → RTURN

(b) Programs induced using NSR.

NSR outperforms Transformer by 23% in HINT.

Model	Symbol Input						Image Input					
	I	SS	LS	SL	LL	Avg.	I	SS	LS	SL	LL	Avg.
GRU	76.2	69.5	42.8	10.5	15.1	42.5	66.7	58.7	33.1	9.4	12.8	35.9
LSTM	92.9	90.9	74.9	12.1	24.3	58.9	83.9	79.7	62.0	11.2	21.0	51.5
Transformer	98.0	96.8	78.2	11.7	22.4	61.5	88.4	86.0	62.5	10.9	19.0	53.1
NeSS	≈0	≈0	≈0	≈0	≈0	≈0	-	-	-	-	-	-
NSR (ours)	**98.0**	**97.3**	**83.7**	**95.9**	**77.6**	**90.1**	**88.5**	**86.2**	**67.1**	**83.2**	**58.2**	**76.0**

The evolution of learned programs is human-like.