Synthesizing Context-free Grammars from Recurrent Neural Networks

迟智名

August 23, 2021

目录

目录

Extract CFGs from RNN

- extracting Automata(DFA sequences) from RNN using L^* algorithm
- DFAS $\rightarrow$ PRSs(pattern rule sets) $\rightarrow$ CFGs

- DFA: $< \Sigma, q_0, Q, F, \delta >$; $\hat{\delta}\,(q_1, wa) = \delta(\hat{\delta}\,(q_1, w))$
- Complete DFA: $\forall (q, a) \in Q \times \Sigma, \delta(q, a)$ is defined
- Sink reject states: Q_R
- $L(A, q_1, q_2) \triangleq \{w \in \Sigma^* \mid \hat{\delta}(q_1, w) = q_2\}$
- defined tokens:$\mathit{def}\,(A, q) \triangleq \{\sigma \in \Sigma \mid \delta(q, \sigma) \notin Q_R\}$
- Set Representation of δ:$S_\delta = \{(q, \sigma, q') \mid \delta(q, \sigma) = q'\}$
- Replacing a State: $\delta_{[q \leftarrow q_n]} : Q' \times \Sigma \to Q'$

Dyck language of order N:$D ::= \epsilon \mid L_i D R_i \mid DD, 1 \leq i \leq N$

- D: Start symbol
- L_i, R_i : matching left and right delimiters
- distance & embedding depth

Regular Expression Dyck language:L_i, R_i derive some regular expression

- Regular Expression :$\{a|b\} \cdot c$
- The Chomsky–Schützenberger representation theorem shows that any context-free language can be expressed as a homomorphic image of a Dyck language intersected with a regular language

Pattern

定义 (Patterns)

A pattern p=$\langle \Sigma, q_0, Q, q_X, \delta \rangle$ is a DFA $A^p = \langle \Sigma, q_0, Q, \{q_X\}, \delta \rangle$ satisfying: $L(A^p) \neq \emptyset$, and either $q_0 = q_X$, or $\operatorname{def}(q_X) = \emptyset$ and $L(A, q_0, q_0) = \{\varepsilon\}$. If $q_0 = q_X$ then p is called circular, otherwise, it is non-circular.

- $L_p = L(p)$
- $p^i = \langle \Sigma, q_0^i, Q^i, q_X^i, \delta^i \rangle$

Composition

定义 (Serial Composition)

Let p^1, p^2 be two non-circular patterns. Their serial composite is the pattern $p^1 \circ p^2 = \langle \Sigma, q_0^1, Q, q_X^2, \delta \rangle$ in which $Q = Q^1 \cup Q^2 \backslash \left\{ q_X^1 \right\}$ and $\delta = \delta^1_{[q_X^1 \leftarrow q_0^2]} \cup \delta^2$. We call q_0^2 the **join state** of this operation.

Composition

定义 (Circular Composition)

Let p^1, p^2 be two non-circular patterns. Their circular composite is the circular pattern $p_1 \circ_c p_2 = \langle \Sigma, q_0^1, Q, q_0^1, \delta \rangle$ in which $Q = Q^1 \cup Q^2 \setminus \{q_X^1, q_X^2\}$ and $\delta = \delta^1_{[q_X^1 \leftarrow q_0^2]} \cup \delta^2_{[q_X^2 \leftarrow q_0^1]}$. We call q_0^2 the join state of this operation.

- $L_p = L_{p_1} \cdot L_{p_2}$
- $L_p = \{L_{p_1} \cdot L_{p_2}\}^*$

Composition

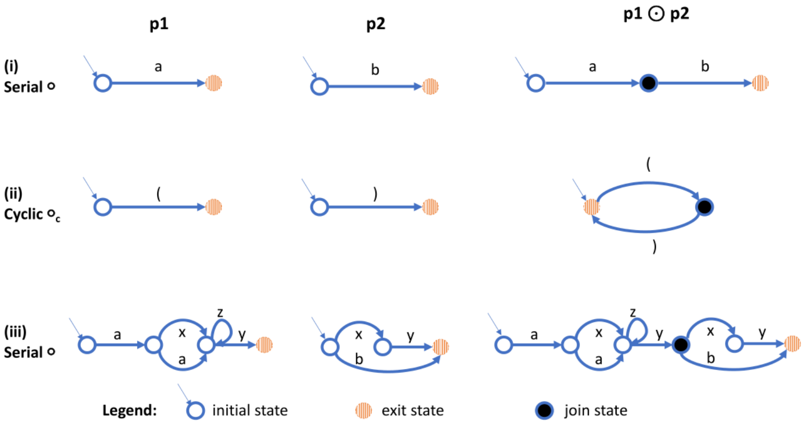


Fig. 2. Examples of the composition operator

Pattern Instances

定义 (Pattern Pair)

A pattern pair is a pair $\langle P, P_c \rangle$ of pattern sets, such that $P_c \subset P$ and for every $p \in P_c$ there exists exactly one pair $p_1, p_2 \in P$ satisfying $p = p_1 \odot p_2$ for some $\odot \in \{\circ, \circ_c\}$. We refer to the patterns $p \in P_c$ as the **composite patterns** of $\langle P, P_c \rangle$, and to the rest as its **base patterns**.

定义 (Pattern Instances)

Let $A = \langle \Sigma, q_0^A, Q^A, F, \delta^A \rangle$ be a DFA, $p = \langle \Sigma, q_0, Q, q_X, \delta \rangle$ be a pattern, and $\hat{p} = \langle \Sigma, q_0', Q', q_X', \delta' \rangle$ be a pattern **inside** A , i.e., $Q' \subseteq Q^A$ and $\delta' \subseteq \delta^A$. We say that $\hat{p}$ is an instance of p in A if $\hat{p}$ is isomorphic to p .

join

定义 (join)

For each composite pattern $p \in P_c$, DFA A, and initial state q of an instance $\hat{p}$ of p in A, $join(p, q, A)$ **returns the join state of** $\hat{p}$ with respect to its composition in $< P, P_c >$.

- A pattern instance $\hat{p}$ in a DFA A is uniquely determined by its structure and initial state: (p, q)

PRS

For infinite DFA sequence $S = \{A_1, A_2, \cdots\}, i \in \mathbb{N}, L(A_i) \subset L(A_{i+1}),\ L(S) = \bigcup_{i=1}^{\infty} L(A_i)$

- May be used to express CFLs,such as $L = \{a^n b^n \mid n \in \mathbb{N}\}$
- infinite $\rightarrow$ finite : finite prefix, noisy; reconstruct the language by guessing how the sequence may continue

Pattern rule sets (PRSs): Create sequences of DFAs with a single accepting state.

- Connect a new pattern instance to the current DFA to a join state of composite patternA_i

PRS

定义 (enabled instances)

An enabled DFA over a pattern pair $\langle P, P_c \rangle$ is a tuple $\langle A, \mathcal{I} \rangle$ such that $A = \langle \Sigma, q_0, Q, F, \delta \rangle$ is a DFA and $\mathcal{I} \subseteq P_c \times Q$ marks **enabled instances** of composite patterns in A.

Given enabled DFA $< A, I >, (p, q) \in I$:

- There is an instance of pattern p in A starting at state q
- We may connect new pattern instances to its join state join(p, q, A).

PRS

定义 (Pattern rule sets)

A PRS **P** is a tuple $\langle \Sigma, P, P_c, R \rangle$ where $\langle P, P_c \rangle$ is a pattern pair over the alphabet Σ and R is a set of rules. Each rule has one of the following forms, for some $p, p^1, p^2, p^3, p^I \in P$, with p^1 and p^2 non-circular: $(1) \perp \rightarrow p^I$

$(2) p \rightarrow_c (p^1 \odot p^2) \propto p^3, \textit{where } p = p^1 \odot p^2 \textit{ for } \odot \in \{o, o_c\}, \textit{and } p^3 \textit{ is circular}$

$(3) p \rightarrow_s (p^1 \circ p^2) \propto p^3, \textit{where } p = p^1 \circ p^2 \textit{ and } p^3 \textit{ is } non-circular$

定义 (Initial Composition)

$\mathcal{D}_1 = \langle A_1, \mathcal{I}_1 \rangle$ is generated from a rule $\perp \rightarrow p^I$ as follows: $A_1 = A^{p^I}$, and $\mathcal{I}_i = \{(p^I, q_0^I)\}$ if $p^I \in P_c$ and otherwise $\mathcal{I}_1 = \emptyset$.

定义 (Rules of type (1))

A rule $\perp \rightarrow p^I$ with circular p^I may extend $\langle A_i, \mathcal{I}_i \rangle$ at the initial state q_0 of A_i. iff $\operatorname{def}(q_0) \cap \operatorname{def}(q_0^I) = \emptyset$. This creates the DFA $A_{i+1} = \left\langle \Sigma, q_0, Q \cup Q^I \backslash \{q_0^I\}, F, \delta \cup \delta^I_{[q_0^I \leftarrow q_0]} \right\rangle$. If $p^I \in P_c$ then $\mathcal{I}_{i+1} = \mathcal{I}_i \cup \{(p^I, q_0)\}$ else $\mathcal{I}_{i+1} = \mathcal{I}_i$.

PRS

定义 (Rules of type (2))

A rule $p \rightarrow_c \left(p^1 \odot p^2\right) \propto p^3$ may extend $\langle A_i, \mathcal{I}_i \rangle$at the join state $q_j = \text{join}\,(p, q, A_i)$ of any instance $(p, q) \in \mathcal{I}_i$, provided $\text{def}\,(q_j) \cap \text{def}\left(q_0^3\right) = \emptyset$. This creates $\langle A_{i+1}, \mathcal{I}_{i+1} \rangle$ as follows:

$A_{i+1} = \left\langle \Sigma, q_0, Q \cup Q^3 \backslash q_0^3, F, \delta \cup \delta^3_{\left[q_0^3 \leftarrow q_j\right]} \right\rangle$, and $\mathcal{I}_{i+1} = \mathcal{I}_i \cup \{(p^k, q^k) \mid p^k \in P_c, k \in \{1, 2, 3\}\}$,

where $q^1 = q$ and $q^2 = q^3 = q_j$

定义 (Rules of type (3))

A rule $p \rightarrow_s \left(p^1 \odot p^2\right) \propto p^3$ may extend $\langle A_i, \mathcal{I}_i \rangle$at the join state $q_j = \text{join}\,(p, q, A_i)$ of any instance $(p, q) \in \mathcal{I}_i$, provided $\text{def}(q_j) \cap \text{def}\left(q_0^3\right) = \emptyset$. This creates $\langle A_{i+1}, \mathcal{I}_{i+1} \rangle$ as follows:

$A_{i+1} = \left\langle \Sigma, q_0, Q \cup Q^3 \backslash q_0^3, F, \delta \cup \delta^3_{[q_0^3 \leftarrow q_j]} \cup C \right\rangle$ where

$C = \{(q_X^3, \sigma, \delta(q_j, \sigma)) \mid \sigma \in \text{def}(p^2, q_0^2)\}$ is **connection transitions**, and

$\mathcal{I}_{i+1} = \mathcal{I}_i \cup \{(p^k, q^k) \mid p^k \in P_c, k \in \{1, 2, 3\}\}$, where $q^1 = q$ and $q^2 = q^3 = q_j$

PRS

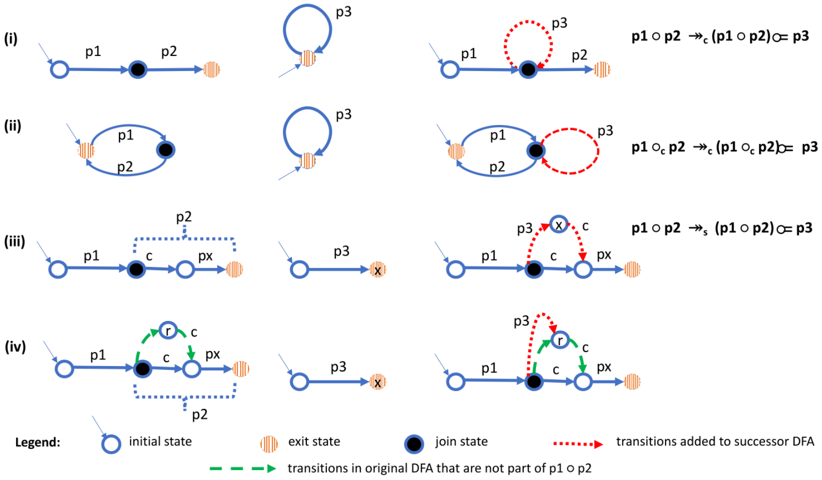
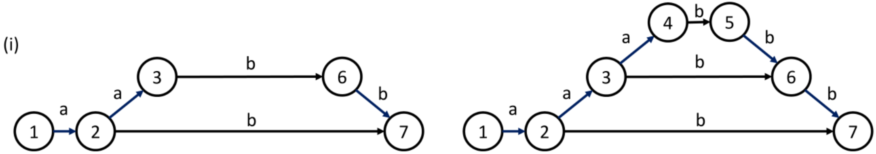


Fig. 3. Structure of DFA after applying rule of type 2 or type 3

Example



- $\perp \rightarrow p^1 \circ p^2$
- $p^1 \circ p^2 \rightarrow_s (p^1 \circ p^2) \odot (p^1 \circ p^2)$

目录

Given DFAs, how to reconstruct PRS $\mathbf{P}$?

Main steps of inference algorithm. Given a sequence of DFAs $A_1 \cdots A_n$, the algorithm infers $\mathbf{P} = \langle \Sigma, P, P_c, R \rangle$ in the following stages:

1. Discover the initial pattern instance $\hat{p}^I$ in A_1. Insert p^I into P and mark $\hat{p}^I$ as enabled. Insert the rule $\perp \rightarrow p^I$ into R.
2. For $i, 1 \leq i \leq n-1$:
   (a) Discover the new pattern instance $\hat{p}^3$ in A_{i+1} that extends A_i.
   (b) If $\hat{p}^3$ starts at the initial state q_0 of A_{i+1}, then it is an application of a rule of type (1). Insert p^3 into P and mark $\hat{p}^3$ as enabled, and add the rule $\perp \twoheadrightarrow p^3$ to R.
   (c) Otherwise ($\hat{p}^3$ does not start at q_0), find the unique enabled pattern $\hat{p} = \hat{p}^1 \odot \hat{p}^2$ in A_i s.t. $\hat{p}^3$'s initial state q is the join state of $\hat{p}$. Add p^1, p^2, and p^3 to P and p to P_c, and mark $\hat{p}^1$,$\hat{p}^2$, and $\hat{p}^3$ as enabled. If $\hat{p}^3$ is non-circular add the rule $p \twoheadrightarrow_s (p^1 \circ p^2) \propto p^3$ to R, otherwise add the rule $p \twoheadrightarrow_c (p^1 \odot p^2) \propto p^3$ to R.
3. Define Σ to be the set of symbols used by the patterns P.

How to Discovering new Patterns

Exit State Discovery algorithm

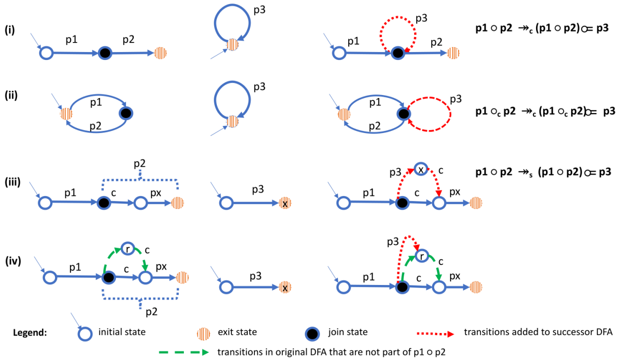
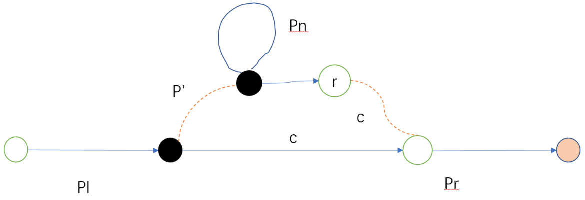


Fig. 3. Structure of DFA after applying rule of type 2 or type 3

Deviations from the PRS framework

- Incorrect pattern creation:threshold
- Simultaneous rule applications



$$p \rightarrow_s (p_l \odot p_r) \propto p'$$
$$p \rightarrow_c (p_1 \odot p_2) \propto p_n$$

algorithm

- $CFG =< \Sigma, N, S, Prod >$: $N, Prod$?
- $\forall p \in P$, $G_p =< \Sigma_p, N_p, Z_p, Prod_p >$
- $P_Y \subseteq P$:LHS of some rule of type(2).
- $N = \{S, C_S, E_S\} \bigcup\limits_{p \in P} \{N_p, E_p\} \bigcup\limits_{p \in P_Y} \{C_p\}$
- $S ::= E_S$, $S ::= C_S E_S$, $C_S ::= C_S C_S$
- For $\perp \to p^I$, $E_S ::= Z_{p^I}$. If circular,$\perp \to p^I$, $C_S ::= Z_{p^I}$
- For each $p \to_c (p^1 \odot p^2) \propto p^3$, $p \to_s (p^1 \circ p^2) \propto p^3$,$Z_p ::= Z_{p_1} E_p Z_{p_2}$, $E_p ::= Z_{p_3}$
- For $p \to_c (p^1 \odot p^2) \propto p^3$,creates $Z_p ::= Z_{p_1} C_p E_p Z_{p_2}$, $C_p ::= C_p C_p$, $C_p ::= Z_{p_3}$
- $Prod = \{ \bigcup\limits_{p \in P} \} \cup Prod'$

目录

- Every RE-Dyck language can be expressed by a PRS.
- But not every CFL can be expressed by a PRS,such as $H = \{a^i x b^i, i \in \mathbb{N}\}$.
- The construction above does not necessarily yield a minimal CFG G equivalent to P.

Experiment setting:

- vote:2
- Sample: weight version of CFG, N=10000
- 2-layer LSTM,hidden dimension = 10, input dimension = 4

LG	DFAs	Init Pats	Final Pats	Min/Max Votes	CFG Correct	LG	DFAs	Init Pats	Final Pats	Min/Max Votes	CFG Correct
L_1	18	1	1	16/16	Correct	L_9	30	6	4	5/8	Correct
L_2	16	1	1	14/14	Correct	L_{10}	6	2	1	3/3	Correct
L_3	14	6	4	2/4	Incorrect	L_{11}	24	6	3	5/12	Incorrect
L_4	8	2	1	5/5	Correct	L_{12}	28	2	2	13/13	Correct
L_5	10	2	1	7/7	Correct	L_{13}	9	6	1	2/2	Correct
L_6	22	9	4	3/16	Incorrect	L_{14}	17	5	2	5/7	Correct
L_7	24	2	2	11/11	Correct	L_{15}	13	6	4	3/6	Incorrect
L_8	22	5	4	2/9	Partial						

Table 1. Results of experiments on DFAs extracted from RNNs

language of $X_n Y_n$:

- $L_1 - L_3$: $(a, b), (a|b, c|d), (ab|cd, ef|gh)$
- $L_3 - L_6$: $(ab, cd), (abc, def), (ab|c, de|f)$

Dyck and RE-Dyck language:

- $L_7 - L_9$:Dyck languages (excluding ϵ) of order 2 through 4
- $L_{10} - L_{11}$: RE-Dyck of order 1,$L_{10}, R_{10} = (abcde, vwxyz), L_{11}, R_{11} = (ab|c, de|f)$

Variations of the Dyck languages:

- L_{12} : alternating single-nested delimiters,([([])]) or [([])]
- $L_{13} - L_{14}$:Dyck-1,2 with additional neutral tokens a,b,c that may appear multiple Times
- L_{15} :Dyck-1,additional neutral tokens abc or d;(abc()())d,a(bc()())d

- Alternating Patterns:L^* extraction had ‘split’ the alternating expressions
- Simultaneous Applications:very large counterexample was returned to L^*:
- Missing Rules: large number of possible delimiter combinations(L_8)
- RNN Noise:d be included between every pair of delimiters in DFAs(L_{15}).

Thank you