APRESENTAÇÃO DE **LEITURA**

Neuro-Symbolic Program Corrector for Introductory Programming Assignments

Sahil Bhatia
Netaji Subhas Institute of Technology
Delhi, India
sahilbhatia.nsit@gmail.com

Pushmeet Kohli
Google Deepmind
London, UK
pushmeet@google.com

Rishabh Singh
Microsoft Research
Redmond, USA
risin@microsoft.com

ALTINO **DANTAS**

2018

I4Soft
Intelligence for Software

UFG

INSTITUTO DE INFORMÁTICA
UFG

Agenda

- Introdução
- Abordagem
- Avalição
- Limitações e trabalhos futuros
- Ameaças à validade
- Resumo

Introdução

- Open Online Courses (MOOCs);
- Geração de feedbacks para submissões;
- Existem várias abordagem que **dependem de AST**;
- Propõe-se uma abordagem híbrida baseada em **RNN** e **sintetização** de código por restrição;
- A ideia é reparar os erros **sintáticos** com a **RNN** e os **funcionais** com um *solver* baseado em **restrição**.

Introdução | Contribuições

- Formalizar a correção de erros sintáticos como problema de **predição** de sequência de *tokens*;
- Apresentar um algoritmo capaz de corrigir erros **sintáticos** e **funcionais**;
- Avaliar a técnica em mais de **14k versões** de programas submetidos por estudantes.

Introdução | Exemplo

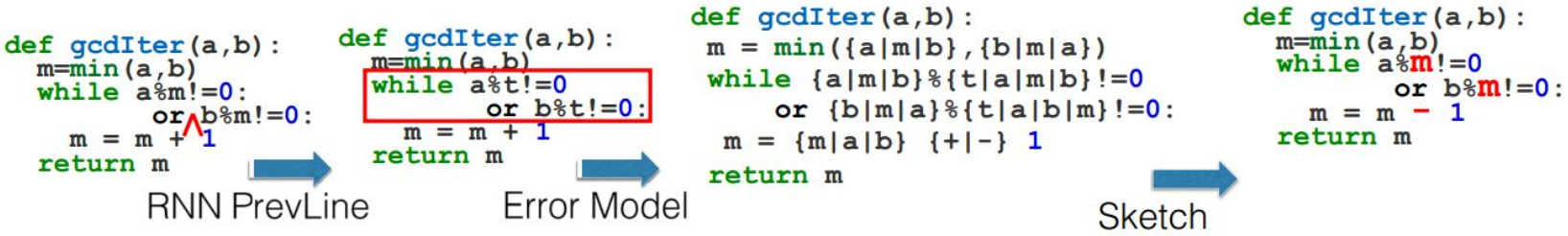


Figure 1: An end-to-end correction example: the RNN model first fixes the syntax error by replacing line 3 but introduces an unknown variable t. The second phase applies an error model to introduce various replacement choices and uses Sketch to find minimal repair for the input program to be functionally correct. An expression of the form {a|b|c} denotes a choice expression, where the SKETCH compiler can choose any expression amongst them. The first option in the choice expression denotes the default expression with cost 0, while all other options result in a cost of 1.

Abordagem

Algoritmo de reparo
neuro-simbólico

Abordagem | Visão geral da proposta

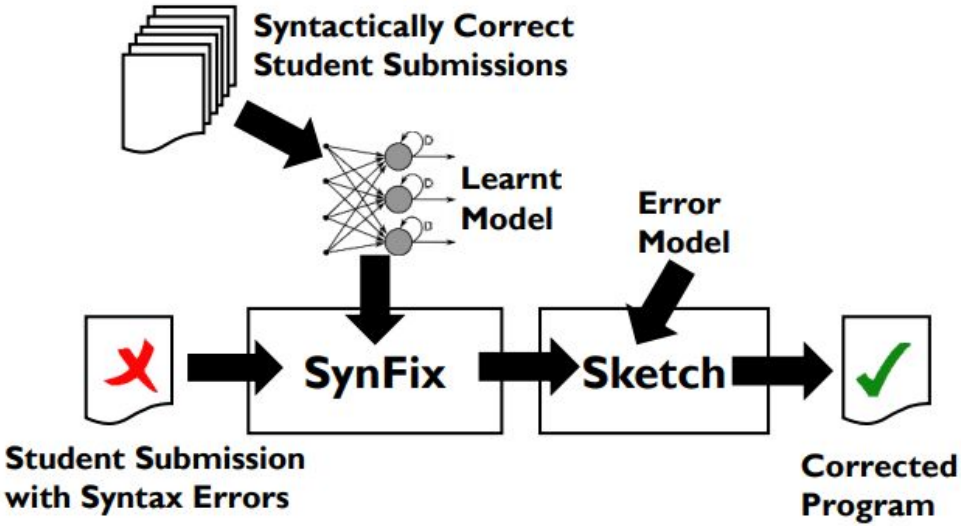


Figure 3: An overview of the system workflow.

Abordagem | Modelo RNN

- Modelo de linguagem baseado em RNN
  - Token IDENT para vocábulos raros;
  - Codificação *One-hot*;
  - Predição de um token depende de uma sequência anterior de *tokens*.

$$h_t = f(W * x_t + V * h_{t-1});$$

$$o_t = \text{softmax}(U * s_t)$$

```
def recPower(b, e):
    if e = 0:
      return 1;
    else:
      return b*recPower(b,e-1)
```

Abordagem | Modelo RNN

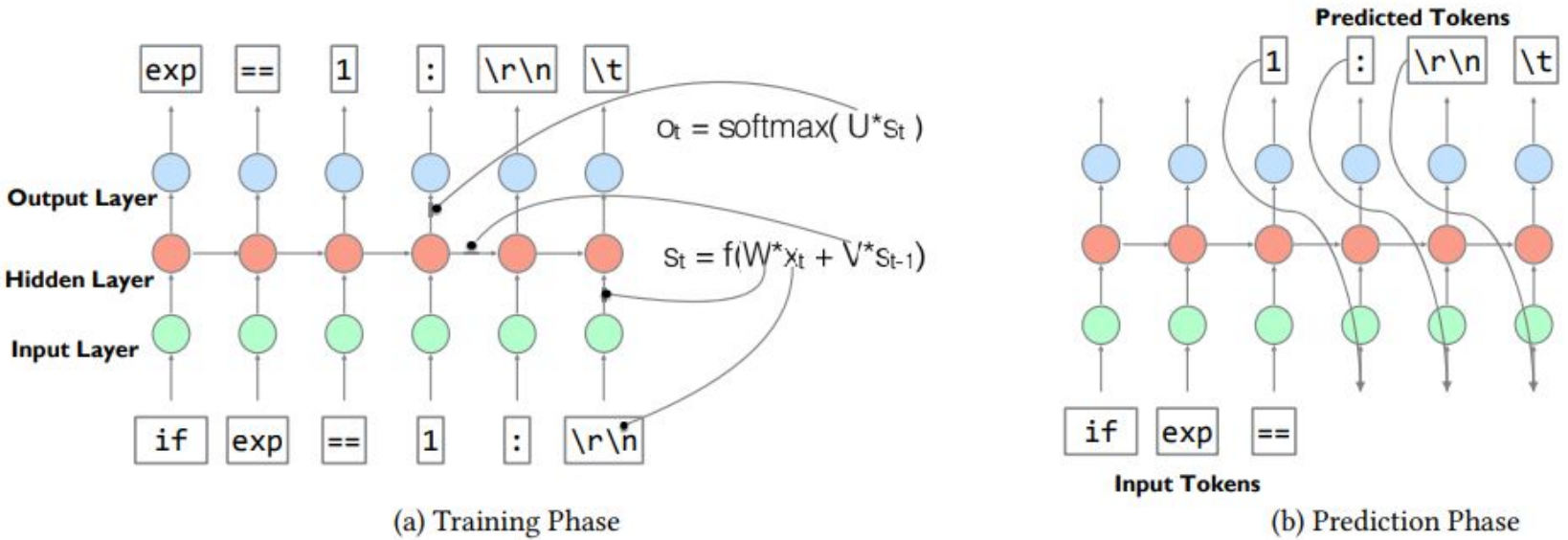


Figure 4: The modeling of our syntax repair problem using an RNN with 1 hidden layer. (a) We provide input and output token sequences in the training phase to learn the weight matrices. (b) In the prediction phase, we provide a token sequence to the input layer of the RNN and generate the output token sequences using the learnt model.

Abordagem | Algoritmo

Algorithm 1 SynFixOne

Input: buggy program P, token sequence model $\mathcal{M}$
(err,loc) := Parse(P); $\widetilde{\mathrm{T}}$:= Tokenize(P)
$\widetilde{\mathrm{T}}_{pref}$:= $\widetilde{\mathrm{T}}$[1..loc]; $\widetilde{\mathrm{T}}_k$:= Predict($\mathcal{M}$, $\widetilde{\mathrm{T}}_{pref}$)
for $i \in$ range$(1, k)$ **do**
  P'_{ins} := Insert(P, loc, $\widetilde{\mathrm{T}}_k[1..i]$)
  if Fixed(P'_{ins}, loc) **return** P'_{ins}
  P'_{repl} := Replace(P, loc, $\widetilde{\mathrm{T}}_k[1..i]$)
  if Fixed(P'_{repl}, loc) **return** P'_{repl}
end for
$\widetilde{\mathrm{T}}^{prev}_{pref}$:= $\widetilde{\mathrm{T}}[1..l_{\mathrm{prev}}(loc)]$; $\widetilde{\mathrm{T}}^{prev}_k$:= Predict($\mathcal{M}$, $\widetilde{\mathrm{T}}^{prev}_{pref}$)
P'_{prev} := ReplaceLine(P, line(loc), $\widetilde{\mathrm{T}}^{prev}_k[1..m]$)
if Fixed(P'_{prev}, , loc) **return** P'_{prev} **else return** ϕ

Algorithm 2 SynFix

Input: buggy program P, sequence model $\mathcal{M}$, max fixes f
for $i \in$ range$(1, f + 1)$ **do**
  P' := SynFixOne(P, $\mathcal{M}$)
  if $P' == \phi$ **return** ϕ
  if Parse(P') = ϕ **return** P'
  else $P := P'$
end for
return ϕ

Abordagem | Reparo semântico por restrição

- 5 regras de tranformação R;
  - $\mathbf{c_{op}} \rightarrow \{< \mid \leq \mid > \mid \geq \mid != \mid ==\}$
  - $\mathbf{a_{op}} \rightarrow \{+ \mid - \mid * \mid / \mid \%\}$
  - **v** → ?v
  - **a** → {a + 1, a - 1, 0, 1}
  - **n** → ??
- Sketch *solver*

$$\exists\, P' \,\forall\, \text{in} : P' = \text{Rewrite}(P, \mathcal{R}) \wedge P_T(in) = P'(in) \wedge \min(\text{Cost}(P, P'))$$

Avaliação

Avaliação | Questões de Pesquisa

1) Quão bem nosso algoritmo é capaz de reparar erros sintáticos e semânticos?

2) Quais opções algorítmicas são escolhidas no algoritmo SynFix para corrigir erros?

3) Como os modelos específicos de problemas se comparam aos modelos gerais?

4) Como o modelo de linguagem RNN se compara com um modelo n-gram?

Avaliação

- 74.818 submissões (programas) consideradas;
- 19,41% com erros sintáticos;
- 9 programas iterativos ou recursivos;
- Fase de treinamento:
  - Tamanho de sequência: 10;
  - Deslocamento: 1;
  - Otimizador **rmsprop** (LR 0,002 / decay 0,97);
  - Duas camadas escondidas com células **LSTM**;
  - **128** unidades cada camada;
  - 50 épocas;

Avalição

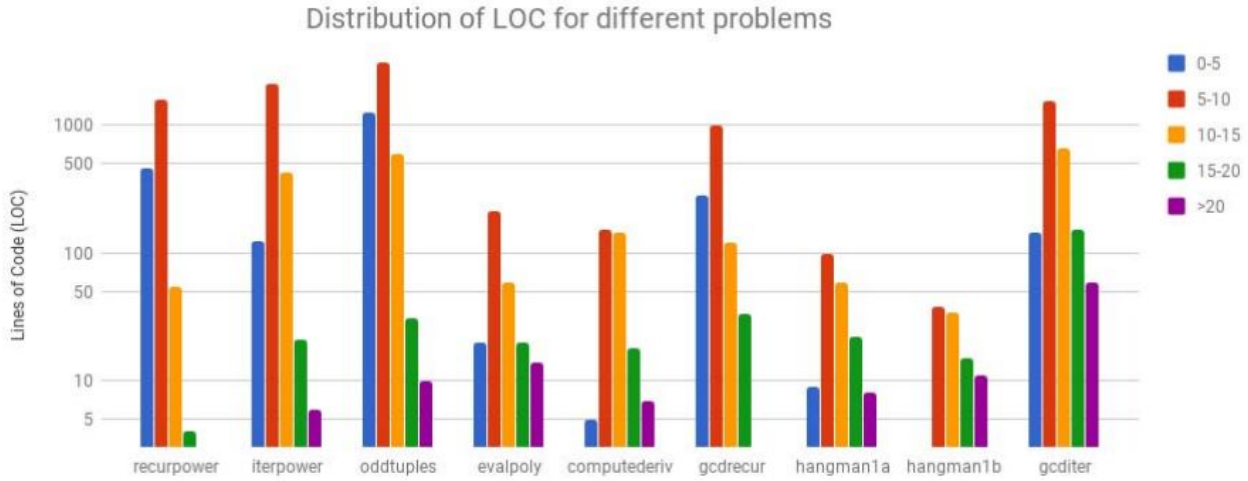


Figure 6: The distribution of lines of code (LOC) for different benchmark problems.

Avaliação | RQ 1

Quão bem nosso algoritmo é capaz de reparar erros sintáticos e semânticos?

Problem	Total Attempts	Syntax Errors (Percentage)	Total Tokens	Training Vocab	Syntax Fixed	Semantic Fixed	RNN + Sketch Semantic Fix
recurPower	10247	2071 (20.21%)	3389858	117	1309 (63.21%)	663 (32.01%)	955 (46.11%)
iterPower	11855	2661 (22.45%)	3558849	526	1864 (70.05%)	401 (15.07%)	667 (25.07%)
oddTuples	29120	4905(16.84%)	1167877	1053	2976 (60.67%)	165 (3.37%)	654 (13.34%)
evalPoly	1148	324 (28.22%)	55370	276	163 (50.31%)	39 (12.04%)	67 (20.68%)
compDeriv	528	323 (61.18%)	18557	150	180 (55.73%)	54 (16.72%)	84 (26.00%)
gcdRecur	7751	1421(18.33%)	275476	274	829 (58.34%)	426 (29.98%)	484 (34.06%)
hangman1	2040	192(9.41%)	106970	398	79 (41.14%)	14 (7.29%)	36 (18.75%)
hangman2	1604	101(6.29%)	108662	546	53 (52.48%)	8 (7.92%)	23 (22.77%)
gcdIter	10525	2528(24.01%)	552405	741	1236 (48.89%)	281(11.15%)	485 (19.18%)
Total	74818	14526(19.4%)	2983124	453	8689(59.8%)	2051(14.1%)	3455(23.8%)

Table 1: The overall results of the SynFix algorithm on 9 benchmark problems.

Avaliação | RQ 2

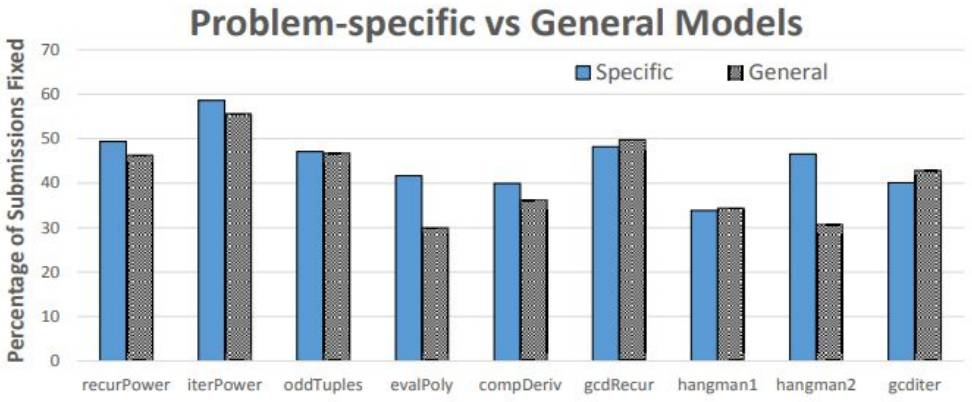
Quais opções algorítmicas são escolhidas no algoritmo SynFix para corrigir erros?

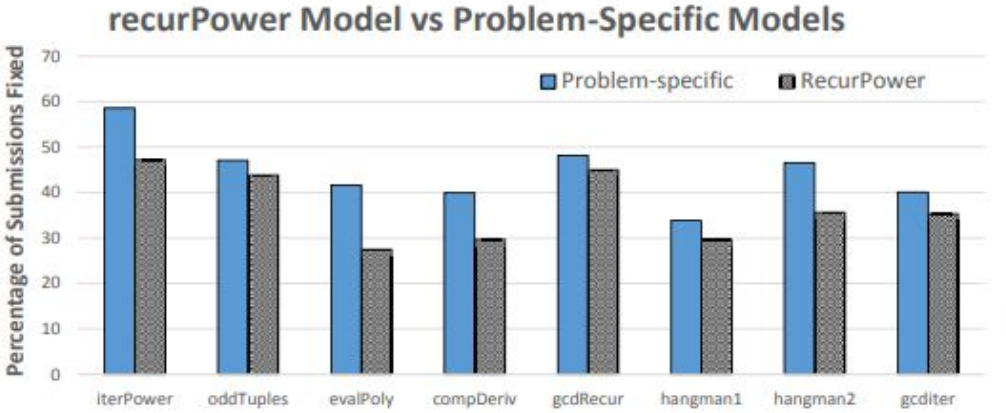
Problem	Incorrect Attempts	Syntactically Fixed					Num. of Fixes	
		Offset		Offset-1		PrevLine	f=1	f=2
		Insert	Replace	Insert	Replace			
recurPower	2071	55	55	574	832	1128	1022	287
iterPower	2661	15	14	746	964	1488	1559	305
oddTuples	4905	198	192	1279	1765	2106	2311	665
evalPoly	324	8	6	49	48	147	135	28
compDeriv	323	2	2	53	80	147	129	51
gcdIter	2528	34	44	637	640	927	1013	223
hangman1	192	1	7	24	43	64	65	14
hangman2	101	2	3	23	18	44	47	6
gcdRecur	1421	20	19	332	620	732	685	144
Total	14526	335	342	3717	5010	6783	6966	1723

Table 2: The detailed breakdown of the algorithmic choices taken by SynFix.

Avaliação | RQ 3

Como os modelos específicos de problemas se comparam aos modelos gerais?





Avaliação | RQ 4

Quais opções algorítmicas são escolhidas no algoritmo SynFix para corrigir erros?

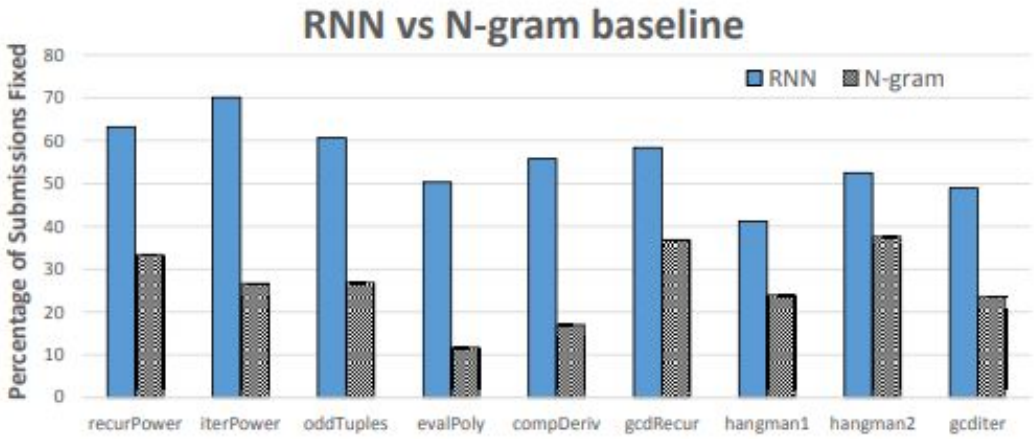


Figure 8: Comparison of the performance of RNN token sequence model vs a 5-gram baseline.

Limitações &
Trabalhos **futuros**

Limitações e trabalhos futuros

Abordagem compara apenas um prefixo para predição;	Pretende-se alterar a RNN para que esta também considere um sufixo;
A proposta atualmente é dependente de um interpretador Python;	Intenta-se desenvolver uma rede capaz de realizar os erros sintáticos;
Necessidade de lidar com a qualidade “pedagógica” do *feedback;*	Deseja-se implementar uma etapa capaz de mensurar quão grave é um dado erro.

Ameaças à validade

Ameaças à validade

- Internas
  - Os reparos podem não ser naturais ou desejáveis;
  - Falta de melhor parametrização da RNN;
- Externa
  - Programas pequenos de uma única linguagem.

Resumo

Introdução | Contribuições

- Formalizar a correção de erros sintáticos como problema de **predição** de sequência de *tokens*;
- Apresentar um algoritmo capaz de corrigir erros **sintáticos** e **funcionais**;
- Avaliar a técnica em mais de **14k versões** de programas submetidos por estudantes.

Abordagem | Visão geral da proposta

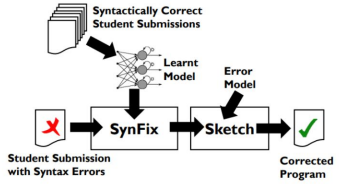


Figure 3: An overview of the system workflow.

Abordagem | Reparo semântico por restrição

- 5 regras de tranformação R;
  - $\mathbf{c}_{op}$ → {< | ≤ | > | ≥ |! = | ==}
  - $\mathbf{a}_{op}$ → {+| - | * |/|%}
  - **v** →?v
  - **a** → {a+1, a-1, 0, 1}
  - **n** → ??
- Sketch *solver*

$$\exists\, P' \,\forall\, \text{in} : P' = \text{Rewrite}(P, \mathcal{R}) \wedge P_T(in) = P'(in) \wedge \min(\text{Cost}(P, P'))$$

Avaliação | RQ 4

Quais opções algorítmicas são escolhidas no algoritmo SynFix para corrigir erros?

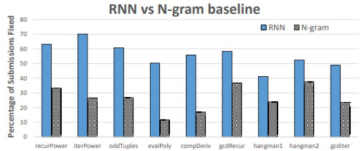


Figure 8: Comparison of the performance of RNN token sequence model vs a 5-gram baseline.

2018 ACM/IEEE 40th International Conference on Software Engineering

Neuro-Symbolic Program Corrector for Introductory Programming Assignments

Sahil Bhatia
Netaji Subhas Institute of Technology
Delhi, India
sahilbhatia.nsit@gmail.com

Pushmeet Kohli
Google Deepmind
London, UK
pushmeet@google.com

Rishabh Singh
Microsoft Research
Redmond, USA
risin@microsoft.com

ABSTRACT

Automatic correction of programs is a challenging problem with numerous real world applications in security, verification, and education. One application that is becoming increasingly important is the correction of student submissions in online courses for providing feedback. Most existing program repair techniques analyze Abstract Syntax Trees (ASTs) of programs, which are unfortunately unavailable for programs with syntax errors. In this paper, we propose a novel Neuro-symbolic approach that combines neural networks with constraint-based reasoning. Specifically, our method first uses a Recurrent Neural Network (RNN) to perform syntax repairs for the buggy programs; subsequently, the resulting syntactically-fixed

1 INTRODUCTION

The increasing importance of computing has resulted in a dramatic growth in the number of students interested in learning programming. Initiatives such as edX and Coursera attempt to meet this demand by providing Massive Open Online Courses (MOOCs) that are easily accessible to students worldwide. While MOOCs have numerous benefits, their effectiveness over the traditional classroom setting is limited by the challenge of providing quality feedback to students on their submissions to open-response programming assignments. We present a learning based technique to automatically generate corrections for student submissions that in turn can be used to generate feedback.

Obrigado

Dúvidas ou sugestões?

altinoneto@inf.ufg.br

i4soft.com.br