Probabilistic Higher Order Grammar: Probabilistic Model for Code

Pavol Bielik, Veselin Raychev, Martin Vechev

ETH Zürich

15 million repositories
Billions of lines of code
High quality, tested, maintained programs

How can we learn probabilistic models directly from data?

The Need for Better Conditioning

Context Free Grammar

\[
a \rightarrow \beta_1 \ldots \beta_n
\]

poor context

Richer context

Condition the predictions on richer context

In general:

Unrestricted programs

(Turing complete)

Our Work:

TCond Language for navigating over trees and accumulating context

Probabilistic Higher Order Grammar (PHOG)

N: set of non-terminal symbols
\( \Sigma \): set of terminal symbols
s: start symbol
C: conditioning set where \( \gamma \subset C \)
\( p \): AST \( \rightarrow \) C
\( q \): R \( \rightarrow \) \( \mathbb{R}^n \)
\( \sum a_q \gamma \rightarrow \beta_1 \ldots \beta_n \) is valid probability distribution

Learn

Existing programs as trees

TCond Language

Search technique

Enumerative search

Genetic programming

\[ p_{\text{best}} = \arg \min_{p \in \text{TCond}} \cos(t(d, p)) + \gamma \cdot Q(p) \]

\[ |d| < |D| \]

Representative sampling

Regularization to avoid too complex programs

Using Programs to Explain Data

PHOG

150k JavaScript Programs

100k: Training Set (L.07 ÷ 10 unordered queries)
50k: Evaluation Set (S.3 ÷ 10 unordered queries)

Evaluation

Code Completion Error Rate

Non-Terminals

Terminals

PCFG

48.5%

49.9%

n-gram

30.8%

28.7%

SVM

46.4%

48.5%

PHOG

32.5%

29.5%

PHOG

25.9%

18.5%

176 unique values

10^4 unique values

10^5 unique values

Code Completion Examples

Identifier 38%
Contains \( \) library

Property 35%
Start = list: length

String 48%
\''\' + attr + \''\'

Number 36%
\[ \cos(x[d(\text{if}(|d|, \text{false}, \text{true}))]) \]

RegExp 34%
line.replace(\text{java}, 'br' , 'br')

UnaryExpr 3%
\text{if}((\text{events})[1] > 2)

BinaryExpr 26%
\text{while}((\text{index})<2)

LogicalExpr 8%
\text{frame} = \text{frame} + 1

Code Completion

Efficient Learning

Trained as efficiently as PCFGs and n-gram models

Widely Applicable

Agnostic to programming language

Flexible Representation

Conditioning for the predictions is determined dynamically

How can we learn probabilistic
models directly from data?