Probabilistic and Interpretable Models of Code

Martin Vechev
ETH Zurich and DeepCode.ai
Learning and probabilistic models based on Big Data have revolutionized entire fields

Natural Language Processing (e.g., machine translation)

Computer Vision (e.g., image captioning)

Medical Computing (e.g., disease prediction)

Can we bring this revolution to programmers?
Machine Learning for Programs

Task → Statistical Programming Tool → Solution

number of repositories

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

last 5 years

Google  facebook  Microsoft
Why now?

**Advances in Programming Languages**
[Automated Reasoning, Synthesis, Constraint Solving]

**Advances in Machine Learning**
[Deep Learning, Graphical Models, Language Models]

**Data**
[> 15 million public repositories]

**Confluence of streams**

**machine learning-based programming tools**
new rules, new ideas, new opportunities
Machine Learning for Programs

Probabilistically likely solutions to problems hard to solve otherwise

Joint work with:

Pavol Bielik  Veselin Raychev  Andreas Krause  Christine Zeller  Svetoslav Karaivanov  Pascal Roos  Benjamin Bichsel  Timon Gehr  Petar Tsankov  Mateo Panzacchi

Publications

- Robust Relational Layout Synthesis from Examples, **ACM OOPSLA’18**
- Inferring Crypto API Rules from Code Changes, **ACM PLDI’18**
- Program Synthesis for Character Level Language Modeling, **ICLR’17**
- Learning a Static Analyzer from Data, **CAV’17**
- V. Raychev, PhD thesis, **ACM Doctoral Dissertation**, Honorable Mention
- Statistical Deobfuscation of Android Applications, **ACM CCS’16**
- Probabilistic Mode for Code with Decision Trees, **ACM OOPSLA’16**
- PHOG: Probabilistic Mode for Code, **ACM ICML’16**
- Learning Programs from Noisy Data, **ACM POPL’16**
- Predicting Program Properties from “Big Code”, **ACM POPL’15, CACM’18**
- Code Completion with Statistical Language Models, **ACM PLDI’14**
- Machine Translation for Programming Languages, **ACM Onward’14**

more: [http://plml.ethz.ch](http://plml.ethz.ch)
Dimensions:
Machine Learning for Programming

- Applications
- Intermediate Representation
- Analyze Program (PL)
- Train Model (ML)
- Query Model (ML)
## Dimensions: Machine Learning for Programming

<table>
<thead>
<tr>
<th>Applications</th>
<th>Code completion</th>
<th>Program synthesis</th>
<th>Feedback generation</th>
<th>Translation</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>Deobfuscation</td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Intermediate Representation</th>
<th>Sequences (sentences)</th>
<th>Translation Table</th>
<th>Graphical Models (CRFs)</th>
<th>Feature Vectors</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>Trees</td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Analyze Program (PL)</th>
<th>typestate analysis</th>
<th>control-flow analysis</th>
<th>alias analysis</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>scope analysis</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Train Model (ML)</th>
<th>Neural Networks</th>
<th>SVM</th>
<th>Structured SVM</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>N-gram/back-off</td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>PCFG</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

| Query Model (ML) | $\text{argmax}_{y \in \Omega} P(y | x)$ | Greedy | MAP inference |
|------------------|---------------------------------------|--------|---------------|
Camera camera = Camera.open();
camera.setDisplayOrientation(90);
camera.unlock();
SurfaceHolder holder = getHolder();
holder.addCallback(this);
holder.setType(SurfaceHolder.STP);
MediaRecorder r = new MediaRecorder();
r.setCamera(camera);
r.setAudioSource(MediaRecorder.AS);
r.setVideoSource(MediaRecorder.VS);
r.setOutFormat(MediaRecorder.MPEG4);

Statistical Code Synthesis

[Code Completion with Statistical Language Models, ACM PLDI 2014
Veselin Raychev, Eran Yahav, M.V.]

Statistical language models
Recurrent neural networks

+ 

Typestate analysis
Alias analysis
Importance of Static (Semantic) Analysis

precision vs. % of data used

- red: no alias analysis
- green: with alias analysis

static analysis benefit = 10x more data
Statistical Language Translation

[ACM Onward’14, Svetoslav Karaivanov, Veselin Raychev, M.V.]

C#  Java
Console.WriteLine("Hi")  System.out.println("Hi");
...
...

Phrase-based Statistical Machine Translation

+  Prefix Parsing of Context-Free Grammars
function FZ(e, t) { var n = []; var r = e.length; var i = 0; for (; i < r; i += t) if (i + t < r) n.push(e.substring(i, i + t)); else n.push(e.substring(i, r)); return n; }

function chunkData(str, step) {
  var colNames = [];
  var len = str.length;
  var i = 0;
  for (; i < len; i += step)
    if (i + step < len)
      colNames.push(str.substring(i, i + step));
    else
      colNames.push(str.substring(i, len));
  return colNames;
}
This Page Amsterdam @thispage_ams · Jul 16
Do you write ugly JavaScript code? Not to worry. JSNice will make it look like you are a superstar coder. Yai! - buff.ly/1HR4JL7

Ingvar Stepanyan @RReverser · Aug 6
JSNice.org became my *must-have* tool for code deobfuscation.

Brevity @seekbrevity · Jul 28
JSNice is an *amazing* tool for de-minifying *javascript* files. JSNice.org, its great for *learning* and reverse engineering.

Alvaro Sanchez @alvasvi · Jun 10
This is gold. Statistical renaming, Type inference and Deobfuscation. jsnice.org

Alex Vanston @mvdot · Jun 7
I've been looking for this for years: JS NICE buff.ly/1pQ5qfr javascript #unminify #deobfuscate #makeItReadable

Kamil Tomšík @cztomsk · Jun 6
tell me *how this* works!
dem-inify *jquery* #javascript incl. args, vars & #jsdoc impressive! jsnice.org
Statistical Program de-obfuscation

[ACM CCS’16, Benjamin Bichsel, Petar Tsankov, Veselin Raychev, M.V.]

```java
package a.b.c
class a extends SQLiteHelper {
    SQLiteDatabase b;
    public a(Context ctx) {
        b = getWritableDatabase();
    }
    Cursor c(String str) {
        return b.rawQuery(str);
    }
}
```

```java
package com.example.dbhelper

class DBHelper extends SQLiteHelper {
    SQLiteDatabase db;
    public DBHelper(Context ctx) {
        db = getWritableDatabase();
    }
    Cursor execSQL(String str) {
        return db.rawQuery(str);
    }
}
```

Funny Reddit post/comment

[-] Tycon712 • 3 points 2 days ago
Can someone tell me what the point of using Proguard is if there are tools out there like this?

[-] theheartbreakpug • 6 points 2 days ago
As far as I know, this is brand new. I asked the creator of ProGuard a week ago how hard it is to unobfuscate code after it's run through proguard. He said it strips all the names out of the code so it's essentially impossible. I'm super impressed by what they've done here.
Nice2Predict.org: scalable structured prediction framework

[Pavol Bielik, Veselin Raychev, Matteo Panzacchi, Nick Baumann, M.V.]

fully, open sourced, Apache license

used by various groups worldwide

Fast, Approximate MAP inference
Fast, Parallel, Structured SVM and Pseudo-Likelihood Training
Arbitrary factors and indicator functions
Fundamental Problem

Data

Learning

Model

Probabilistic Model

Widely Applicable
Efficient Learning
High Precision
Explainable Predictions
Training dataset $D$

```plaintext
f.open("f2" | "r");
f.read();

f.open("f2" | "w");
f.write("c");

f.open("f1" | "r");
f.read();
```

query:

```plaintext
f.open("file" | "r");
```
Training dataset $D$

```c
f.open("f2" | "r");
f.read();

f.open("f2" | "w");
f.write("c");

f.open("f1" | "r");
f.read();
```

query:

```c
f.open("file" | "r");
```

3-gram model on tokens

Hindle et. al., ACM ICSE’12

<table>
<thead>
<tr>
<th>Action</th>
<th>Context</th>
<th>Probability</th>
</tr>
</thead>
<tbody>
<tr>
<td>P(open $</td>
<td>$ f.)</td>
<td>$\gamma$</td>
</tr>
<tr>
<td>P(read $</td>
<td>$ f.)</td>
<td>$\gamma$</td>
</tr>
<tr>
<td>P(write $</td>
<td>$ f.)</td>
<td>$\gamma$</td>
</tr>
</tbody>
</table>
Training dataset $D$

```python
f.open("f2" | "r");
f.read();

f.open("f2" | "w");
f.write("c");

f.open("f1" | "r");
f.read();
```

query:

```python
f.open("file" | "r");
```

3-gram model on tokens

Hindle et. al., ACM ICSE’12

\[
P(open | f.) ~ 3/6
P(read | f.) ~ 2/6
P(write | f.) ~ 1/6
\]

context $\gamma$
Training dataset $D$

```python
f.open("f2" | "r");
f.read();

f.open("f2" | "w");
f.write("c");

f.open("f1" | "r");
f.read();
```

**probabilistic model on APIs**

Raychev et. al., ACM PLDI’14

- $P(\text{read} \mid \text{open}) \sim 2/3$
- $P(\text{write} \mid \text{open}) \sim 1/3$

query:

```python
f.open("file" | "r");
```

**3-gram model on tokens**

Hindle et. al., ACM ICSE’12

- $P(\text{open} \mid f.) \sim 3/6$
- $P(\text{read} \mid f.) \sim 2/6$
- $P(\text{write} \mid f.) \sim 1/6$
Training dataset $D$

```python
f.open("f2" | "r");
f.read();

f.open("f2" | "w");
f.write("c");

f.open("f1" | "r");
f.read();
```

probabilistic model on APIs
Raychev et. al., ACM PLDI’14

\[
P(\text{read} \mid \text{open}) \sim \frac{2}{3}, \quad P(\text{write} \mid \text{open}) \sim \frac{1}{3}
\]

query:

```python
f.open("file" | "r");
f.\textbf{read}
```

3-gram model on tokens
Hindle et. al., ACM ICSE’12

\[
P(\text{open} \mid f.) \sim \frac{3}{6}, \quad P(\text{read} \mid f.) \sim \frac{2}{6}, \quad P(\text{write} \mid f.) \sim \frac{1}{6}
\]
Training dataset $D$

```python
f.open("f2" | "r");
f.read();

f.open("f2" | "w");
f.write("c");

f.open("f1" | "r");
f.read();
```

probabilistic model on APIs

Raychev et. al., ACM PLDI’14

$$P(\text{read} \mid \text{open}) \sim \frac{2}{3}$$

$$P(\text{write} \mid \text{open}) \sim \frac{1}{3}$$

query:

```python
f.open("file" | "r");
f.?
```

3-gram model on tokens

Hindle et. al., ACM ICSE’12

$$P(\text{open} \mid \text{f.}) \sim \frac{3}{6}$$

$$P(\text{read} \mid \text{f.}) \sim \frac{2}{6}$$

$$P(\text{write} \mid \text{f.}) \sim \frac{1}{6}$$

What should the context be?
“...All problems in computer science can be solved by another level of indirection...”
-- David Wheeler
“...All problems in computer science can be solved by another level of indirection...”

-- David Wheeler

key idea: synthesize a function $f$: $\frac{\text{program}}{\text{context}} \rightarrow \gamma$
Creating probabilistic models: our method


1. **Pick** a structure of interest, e.g., ASTs:

2. **Define** a DSL for expressing functions: (can be Turing complete)

3. **Synthesize** \( f_{\text{best}} \in \text{DSL} \) from Dataset \( D \):

\[
 f_{\text{best}} = \arg\min_{f \in \text{DSL}} \text{cost}(D, f)
\]

4. **Use** \( f_{\text{best}} \) to compute context and predict:

\[
 f_{\text{best}} \left( \begin{array}{c} \text{} \\ \text{} \end{array} \right) \rightarrow \gamma \]
Step 1: Pick Structure of Interest

Let it be abstract syntax trees (ASTs) of programs

**JavaScript program**

elem.notify({
    position: 'top',
    autoHide: false,
    delay: 100
});

**AST**
Step 2: Define a DSL over structure

Syntax

\[
T\text{Cond} ::= \varepsilon \mid \text{WriteOp } T\text{Cond} \mid \text{MoveOp } T\text{Cond}
\]

\[
\text{MoveOp} ::= \text{Up, Left, Right, DownFirst, DownLast, NextDFS, PrevDFS, NextLeaf, PrevLeaf, PrevNodeType, PrevNodeValue, PrevNodeContext}
\]

\[
\text{WriteOp} ::= \text{WriteValue, WriteType, WritePos}
\]

Semantics

<table>
<thead>
<tr>
<th>Up</th>
<th>Left</th>
</tr>
</thead>
<tbody>
<tr>
<td>WriteValue</td>
<td></td>
</tr>
</tbody>
</table>

\[
\gamma \leftarrow \gamma \cdot \Box
\]
Step 3: synthesize $f_{\text{best}}$

$$f_{\text{best}} = \operatorname{argmin} \text{cost}(D, f)$$
$$f \in \text{DSL}$$
Step 3: synthesize $f_{best}$

**DSL**

```
TCond ::= $f$ | WriteOp TCond | MoveOp TCond
MoveOp ::= Up, Left, Right, ...
WriteOp ::= WriteValue, WriteType, ...
```

**Synthesizer**

- **Generate candidate $f$**

  \[ f_{best} = \text{argmin} \text{ cost}(D, f) \quad f \in \text{DSL} \]

**Build Probabilistic Model $P$**

- **Use $D$ and $f$ to compute**

  \[ \text{cost}(D, f) = \text{entropy}(P) \]

- **Dataset $D$**

  millions ($\approx 10^8$)

- **$O(|D|)$**

- **To scale: iterative synthesis on fraction of examples**
Step 4: use $f_{best}$ to predict

program

```javascript
elem.notify(
  ...
  ...
  {
    position: 'top',
    hide: false,
    ?
  }
)
```

$f_{best}$

```javascript
Left
WriteValue
Up
WritePos
Up
DownFirst
DownLast
WriteValue
```

Context $\gamma$

```javascript
{}
{hide}
{hide}
{hide, 3}
{hide, 3}
{hide, 3}
{hide, 3}
```

\{Previous Property, Parameter Position, API name\}
Deep3: Experimental Results
[Probabilistic Model of JavaScript]

Dataset $D$: 150,000 files  
Training Time: $\sim 100$ hours  
$f_{best} \sim 50,000$ instr.

<table>
<thead>
<tr>
<th>Probabilistic Model</th>
<th>Accuracy [APIs]</th>
</tr>
</thead>
<tbody>
<tr>
<td>Last two tokens, Hindle et. al. [ICSE’12]</td>
<td>22.2%</td>
</tr>
<tr>
<td>Last two APIs, Raychev et. al. [PLDI’14]</td>
<td>30.4%</td>
</tr>
<tr>
<td><strong>Deep3</strong></td>
<td><strong>66.6%</strong></td>
</tr>
</tbody>
</table>

Details in: “Probabilistic Model for Code with Decision Trees”, ACM OOPSLA’16
Applying the Concept to Natural Language
[Program Synthesis for Character Level Language Modeling, ICLR’17]

Dataset D: Hutter Prize Wikipedia Dataset

Training Time: ~8 hours

$f_{\text{best}} \sim 9,000$ instr

Uses a char-level DSL with state

Interpretable model, browse here: http://www.srl.inf.ethz.ch/charmmodel.html

<table>
<thead>
<tr>
<th>Probabilistic Model</th>
<th>Bits-per-Character</th>
</tr>
</thead>
<tbody>
<tr>
<td>7-gram (best)</td>
<td>1.94</td>
</tr>
<tr>
<td>Stacked LSTM (Graves 2013)</td>
<td>1.67</td>
</tr>
<tr>
<td>Char-based DSL synthesis</td>
<td>1.62</td>
</tr>
<tr>
<td>MRNN (Sutskever 2011)</td>
<td>1.60</td>
</tr>
<tr>
<td>MI-LSTM (Wu et al. 2016)</td>
<td>1.44</td>
</tr>
<tr>
<td>HM-LSTM* (Chung et al. 2016)</td>
<td>1.40</td>
</tr>
</tbody>
</table>
More Resources

Learning from Large Codebases,
ACM Doctoral Dissertation, Honorable Mention Award

- http://plml.ethz.ch
- Dagstuhl Seminar on Big Code Analytics, Nov 2015
- Data sets, tools, challenges: http://learningfrombigcode.org

Veselin Raychev
1. Pick a structure of interest, e.g., trees:

2. Define a DSL for expressing functions: (can be Turing complete)

3. Synthesize $f_{\text{best}} \in \text{DSL}$ from Dataset $D$:

$$f_{\text{best}} = \arg\min_{f \in \text{DSL}} \text{cost}(D, f)$$

4. Use $f_{\text{best}}$ on new structures:

$$f_{\text{best}} (\quad ) \rightarrow \gamma$$

more: [http://plml.ethz.ch](http://plml.ethz.ch)