Programming with “Big Code”: Lessons, Techniques, Applications

Pavol Bielik, Veselin Raychev, Martin Vechev
Department of Computer Science
ETH Zurich
Work on “Big Code” started a few years ago

**Prof. Martin Vechev**
**Prof. Andreas Krause**
**Veselin Raychev**
**Pavol Bielik**
**Svetoslav Karaivanov**
**Christine Zeller**
**Pascal Roos**

**Code Completion with Statistical Language Models**, PLDI 2014
**Machine Translation for Programming Languages**, Onward 2014
**Predicting Program Properties from “Big Code”**, POPL 2015
**Fast and Precise Statistical Code Completion**, ETH TR
**Statistical Feedback Generation for Programs**, ETH TR
**Programming with Big Code: Lessons, Techniques and Applications**, SNAPL 2015
All of these benefit from the “Big Code” and lead to applications not possible with previous techniques.
All of these benefit from the “Big Code” and lead to applications not possible with previous techniques.
Applications

[PLDI 14]
SLANG: Code Completion

```
Intent i = new Intent();
    ?
ctx.sendBroadcast(i);
```

[Onward 14]
Programming Language Translation

```
P( Java | C# )
P( C# | Java )
P( Java )
```

[submitted]
Statistical Feedback Generation

```
... for x in range(a):
    print a[x]
```

likely error

All of these benefit from the “Big Code” and lead to applications not possible with previous techniques.
Applications

[PLDI 14]
SLANG: Code Completion

```java
Intent i = new Intent();
ctx.sendBroadcast(i);
```

[Onward 14]
Programming Language Translation

```
P( Java | C# )
P( C# | Java )
P( Java )
```

[POPL 15]
JSNice: Deobfuscation
Type Prediction

[submitted]
Statistical Feedback Generation

```
... for x in range(a):
    print a[x]
```

All of these benefit from the “Big Code” and lead to applications not possible with previous techniques.
Probabilistic Programming Systems: Dimensions

Applications

Intermediate Representation

Analyze Program (PL)

Train Model (ML)

Query Model (ML)
What is a generic metric for code?

- ✔ Cross Entropy → ✔ Code Completion
- ✔ BLEU Score → ✔ Program Translation

Traditional metrics might not be indicative of client performance.
What is the best program representation?
What is the best program representation?

**Sequences**

- req → \{<open, 0>, <send, 0>\}
- source → \{..., <open, 2>\}

**Trees**

```
  =
 a
   +
 x
   y
```

**Graphical Models**

```
  node
  node
  node
  node
```

**Feature Vectors**

- req → (0,0,1,1,0)
- source → (1,0,0,0,0)
- ...

**Probabilistic Programming Systems: Dimensions**
Probabilistic Programming Systems: Dimensions

What is the best program representation?

Choosing the right representation is crucial

Feedback Generation: Sequence representations

<table>
<thead>
<tr>
<th>Representation</th>
<th>Percentage</th>
</tr>
</thead>
<tbody>
<tr>
<td>Allamanis et. al. [2013]</td>
<td>46.4%</td>
</tr>
<tr>
<td>Hsiao et. al. [2014]</td>
<td>50.8%</td>
</tr>
<tr>
<td>Incorporate semantic information</td>
<td>75.3%</td>
</tr>
<tr>
<td>Incorporate dataflow analysis</td>
<td>86.3%</td>
</tr>
</tbody>
</table>

Applications

Intermediate Representation

Analyze Program (PL)

Train Model (ML)

Query Model (ML)
How to extract program representation?

SLANG (APIs): alias and typestate analysis

JSNice (Variable Names): scope and alias analysis

Feedback Generation: alias, control-flow and typestate analysis

```c
req.open("GET", source, false);
```

```plaintext
req → {<open, 0>, <send, 0>}
source → {..., <open, 2>}
```
How to extract program representation?

SLANG (APIs): alias and typestate analysis
JSNice (Variable Names): scope and alias analysis
Feedback Generation: alias, control-flow and typestate analysis

Design scalable yet precise enough algorithms

- [Precision vs % of data used]

- 1%: no alias analysis
- 10%: no alias analysis
- 100%: no alias analysis

- 1%: with alias analysis
- 10%: with alias analysis
- 100%: with alias analysis
What is the suitable probabilistic model?

- N-gram language model
- Probabilistic context-free grammars
- Neural networks
- (Structured) Support vector machine
- Conditional Random Fields
- ...
What is the suitable probabilistic model?

- N-gram language model
- Probabilistic context-free grammars
- Neural networks
- (Structured) Support vector machine
- Conditional Random Fields

Structured prediction is critical

<p>| | |</p>
<table>
<thead>
<tr>
<th></th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>Baseline</td>
<td>25.3%</td>
</tr>
<tr>
<td>Independent</td>
<td>54.1%</td>
</tr>
<tr>
<td>Structured</td>
<td>63.4%</td>
</tr>
</tbody>
</table>
## Programming with “Big Code”

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

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

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

| Query Model                | $\text{argmax } P(y \mid x)$ | $y \in \Omega$        | |

Programming with “Big Code”

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

| Intermediate Representation   | Sequences (sentences)            | Translation Table                                                   |                                      |
|                               | Trees                            |                                                                     |                                      |

| Analyze Program (PL)          | alias analysis                   | control-flow analysis                                               |                                      |
|                               | scope analysis                   | typestate analysis                                                  |                                      |

| Train Model (ML)              | Neural Networks                  | SVM                                                                 |                                      |
|                               | N-gram language model            |                                                                     |                                      |

| Query Model                  | \[ \text{argmax} \ P(y \mid x) \] | \( y \in \Omega \)                                                 |                                      |

We have open-sourced our prediction engine and we are extending it with new capabilities.

Upcoming PLDI’15 tutorial
Programming with “Big Code”

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

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

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

<table>
<thead>
<tr>
<th>Train Model (ML)</th>
<th>Neural Networks</th>
<th>SVM</th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>N-gram language model</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Query Model</th>
<th>argmax ( P(y \mid x) )</th>
<th>Greedy MAP Inference</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>( y \in \Omega )</td>
<td></td>
</tr>
</tbody>
</table>

More information and tutorials at: