Learning from “Big Code”

21th PeWe, April 7, 2017

Pavol Bielik
Software Reliability Lab
Department of Computer Science
ETH Zurich
Big Data Impact

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

Computer Vision
(e.g., image captioning)

Medical Computing
(e.g., disease prediction)
Big Data Impact

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?
Vision

Create new kinds of software tools that leverage massive codebases to solve problems beyond what is possible with traditional techniques.

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

last 8 years
Statistical Software Tools

**Write new code:**
Code Completion

Camera camera = Camera.open();
camera.SetDisplayOrientation(90);

**Port code:**
Programming Language Translation

**Understand code/security:**
JavaScript Deobfuscation
Type Prediction

**Debug code:**
Statistical Bug Detection

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.
Understand code/security:
JavaScript Deobfuscation
Type Prediction

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

Debug code:
Statistical Bug Detection

Port code:
Programming Language Translation

Write new code:
Code Completion

Camera camera = Camera.open();
camera.SetDisplayOrientation(90);

...
Code understanding/security

+ JavaScript
Code understanding/security

savePassword(user, password) $\rightarrow$ Obfuscation/Minification $\rightarrow$ B(d, c)
Code understanding/security

savePassword(user, password) $\rightarrow$ Obfuscation/Minification $\rightarrow$ B(d, c)
Code understanding/security

savePassword(user, password) → Obfuscation/Minification → B(d, c)
Code understanding/security

savePassword(user, password) → B(d, c)

savePassword(user, password) ← B(d, c)
Code understanding/security

savePassword(user, password) → Obfuscation/Minification → B(d, c)

savePassword(user, password) ← B(d, c)

Security Analyst
available online:

www.jsnice.org
Impact

30,000 Users in 1st week
Impact

30,000 Users in 1st week
Impact

30,000 Users in 1st week

50 fantastic freebies for web designers, July 2014

By Juan Pablo Sarmiento

CodeGeekz

20 Essential Tools for Coders

14 Essential Tools for Programmers
Everyone knows very well, that when it comes to web development than coding is the most important part where every web developer spends it ...
How to build such tools?

Write new code:
Code Completion

```java
Camera camera = Camera.open();
camera.SetDisplayOrientation(90);
?
```

Port code:
Programming Language Translation

```
C#
Console.WriteLine("Hi");
...

Java
System.out.println("Hi");
...
```

Understand code/security:
JavaScript Deobfuscation
Type Prediction

Debug code:
Statistical Bug Detection

```python
... for x in range(a):
    print a[x]
likely error
```
Applications

```javascript
function get(a, b, c) {
    b.open("GET", a, false);
    b.send(c);
}
```

```javascript
function get(url, client, data) {
    client.open("GET", url, false);
    client.send(data);
}
```

Predicting types

Predicting names

// @param {string} url
// @param {Object} client
// @param {string} data
What is a suitable program representation?

Natural Language Processing

The dog saw a man in the park

Programming Languages

JavaScript  →  ?

Java
What is a suitable program representation?

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

### Trees
- req → (0,0,1,1,0)
- source → (1,0,0,0,0)
- ...
What is a suitable program representation?

**Sequences**

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

**Trees**

- a = +
  - x
  - y

**Graphical Models**

**Feature Vectors**

- req → (0,0,1,1,0)
- source → (1,0,0,0,0)
- ...
function get(a, b, c) {
    b.open("GET", a, false);
    b.send(c);
}
function get(a, b, c) {
    b.open("GET", a, false);
    b.send(c);
}
function get(a, b, c) {
    b.open("GET", a, false);
    b.send(c);
}
function get(a, b, c) {
  b.open("GET", a, false);
  b.send(c);
}
function get(a, b, c) {
  b.open("GET", a, false);
  b.send(c);
}
function get(a, b, c) {
    b.open("GET", a, false);
    b.send(c);
}
function get(a, b, c) {
    b.open("GET", a, false);
    b.send(c);
}

Analyze Program (PL)

Intermediate Representation

Applications

JS NICE Representation
```javascript
function get(a, b, c) {
    b.open("GET", a, false);
    b.send(c);
}
```
function get(a, b, c) {
    b.open("GET", a, false);
    b.send(c);
}

Call graph analysis
Who are the callers of get function?

Applications

Scope analysis
Program location where b was defined?

Intermediate Representation

Typestate analysis
State of the object b?

Alias analysis
b and c point to the same object?

Analyze Program (PL)
Importance of Program Analysis

![Precision vs % of data used]

- **Applications**
- **Intermediate Representation**
- **Analyze Program (PL)**

- **no alias analysis**
- **with alias analysis**

- Precision vs % of data used:
  - 0%
  - 50%
  - 100%
Undirected graphical model
Captures dependencies between facts to be predicted

\[
P(\text{get}, c, a) = \varphi_1(a, c) \times \varphi_2(a, \text{get}) \times \varphi_2(c, \text{get}) \times \ldots / Z(\text{get}, c, a)
\]

makes \( P \) a valid probability distribution

very expensive to compute
Conditional Random Fields
(J. Lafferty, A. McCallum, F. Pereira, ICML 2001)

Undirected graphical model

Captures **dependencies** between facts to be predicted

Captures **conditional distribution** on known facts

\[
P(Y \mid X) = \phi_1(a, c) \times \phi_2(a, \text{get}) \times \phi_2(c, \text{get}) \times ... / Z(Y)
\]

makes \( P \) a valid probability distribution
very expensive to compute
MAP inference example

\[
\arg\max_Y P(Y|X)
\]

```
b.open("GET", a, false);
```
MAP inference example

\[
\text{argmax}_y P(Y|X)
\]

```
b.open("GET", a, false);
```
MAP inference example

\[
\text{argmax}_{c,t} \; P(c,t|v=\text{open})
\]

<table>
<thead>
<tr>
<th>v</th>
<th>c</th>
<th>Score</th>
</tr>
</thead>
<tbody>
<tr>
<td>open</td>
<td>req</td>
<td>6</td>
</tr>
<tr>
<td>open</td>
<td>client</td>
<td>5</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>t</th>
<th>c</th>
<th>Score</th>
</tr>
</thead>
<tbody>
<tr>
<td>client</td>
<td>url</td>
<td>8</td>
</tr>
<tr>
<td>client</td>
<td>link</td>
<td>5</td>
</tr>
<tr>
<td>req</td>
<td>link</td>
<td>2</td>
</tr>
</tbody>
</table>

b.open("GET", a, false);
**MAP inference example**

$$\text{argmax}_{c,t} P(c,t|v=\text{open})$$

### Table 1: \(\varphi_1\)

<table>
<thead>
<tr>
<th>v</th>
<th>c</th>
<th>Score</th>
</tr>
</thead>
<tbody>
<tr>
<td>open</td>
<td>req</td>
<td>6</td>
</tr>
<tr>
<td>open</td>
<td>client</td>
<td>5</td>
</tr>
</tbody>
</table>

### Table 2: \(\varphi_2\)

<table>
<thead>
<tr>
<th>v</th>
<th>t</th>
<th>Score</th>
</tr>
</thead>
<tbody>
<tr>
<td>open</td>
<td>link</td>
<td>7</td>
</tr>
<tr>
<td>open</td>
<td>url</td>
<td>5</td>
</tr>
<tr>
<td>open</td>
<td>address</td>
<td>2</td>
</tr>
</tbody>
</table>

Maximize product of scores: 

$$6 \times 7 \times 2 = 84$$

```java
b.open("GET", a, false);
```
MAP inference example

\[ \text{argmax}_{c,t} \ P(c,t|v=\text{open}) \]

\[
\begin{array}{|c|c|c|}
\hline
v & c & \text{Score} \\
\hline
\text{open} & \text{req} & 6 \\
\text{open} & \text{client} & 5 \\
\hline
\end{array}
\]

Maximize product of scores:
\[ 5 \times 7 \times 5 = 175 \]

\[
\begin{array}{|c|c|c|}
\hline
v & t & \text{Score} \\
\hline
\text{open} & \text{link} & 7 \\
\text{open} & \text{url} & 5 \\
\hline
\end{array}
\]

\[
\begin{array}{|c|c|c|}
\hline
t & c & \text{Score} \\
\hline
\text{client} & \text{url} & 8 \\
\text{client} & \text{link} & 5 \\
\text{req} & \text{link} & 2 \\
\hline
\end{array}
\]

b.open("GET", a, false);
MAP inference example

\[ \text{argmax}_{c,t} \ P(c,t|v=\text{open}) \]

Maximize product of scores:
\[ 5 \times 5 \times 8 = 200 \]

\begin{tabular}{|c|c|c|}
\hline
\textbf{v} & \textbf{c} & \textbf{Score} \\
\hline
open & req & 6 \\
open & client & 5 \\
\hline
\end{tabular}

\begin{tabular}{|c|c|c|}
\hline
\textbf{t} & \textbf{c} & \textbf{Score} \\
\hline
client & url & 8 \\
client & link & 5 \\
req & link & 2 \\
\hline
\end{tabular}

b.open("GET", a, false);
Our goal is to find the most likely assignment of $y$ that satisfies the constraints, also known as MAP inference:

$$y = \arg\max_y P(y'|x) = \arg\max_y \frac{1}{Z} \prod \phi_i(x,y)$$

**Good news:**
the expensive partition function $Z(x)$ is unnecessary
Our goal is to find the most likely assignment of $y$ that satisfies the constraints, also known as **MAP inference**:

$$y = \arg\max_y P(y'|x) = \arg\max_y \frac{1}{Z} \prod \phi_i(x,y)$$

**Good news:**
the expensive partition function $Z(x)$ is unnecessary

**Bad news:**
computing the argmax is still NP-hard (Max-SAT)
Our goal is to find the most likely assignment of $y$ that satisfies the constraints, also known as **MAP inference**:

$$y = \text{argmax}_y, P(y'|x) = \text{argmax}_y, \frac{1}{Z} \prod \phi_i(x,y)$$

**Good news:**
many approximate algorithm exists (Variational Methods, EM, Gibbs sampling, Elimination Algorithm, Junction-Tree algorithm)
Our goal is to find the most likely assignment of $y$ that satisfies the constraints, also known as **MAP inference**:

$$y = \text{argmax}_y \ P(y'|x) = \text{argmax}_y \ 1/Z \ \prod \phi_i(x,y)$$

**Good news:**
many approximate algorithm exists (Variational Methods, EM, Gibbs sampling, Elimination Algorithm, Junction-Tree algorithm)

**Bad news:**
still too slow for learning
Our goal is to find the most likely assignment of $y$ that satisfies the constraints, also known as **MAP inference**:

$$y = \arg\max_y P(y' | x) = \arg\max_y \frac{1}{Z} \prod \phi_i(x, y)$$

**Good news:**
approximate algorithms designed to fit our setting
Learning

\[
P(y|x) = \frac{1}{Z} \exp \sum \lambda_i f_i(x,y)
\]

Learning finds weights \( \lambda_i \) from training data

\[
D = \{ x^{(j)}, y^{(j)} \}_{j=1..n}
\]

programs with facts of interest already manually annotated

Big codebase to learn from
Programmers have spent countless hours to develop, maintain and annotate
Structured SVM

Generalizes SVM, learns weights such that:

$$\forall j \forall y \sum \lambda_i f_i(x^{(j)}, y^{(j)}) \geq \sum \lambda_i f_i(x^{(j)}, y) + \Delta(y, y^{(j)})$$

for all training data samples

the given prediction is better than any other prediction by at least a margin

Training procedure:

N.Ratliff, J. Bagnell, M. Zinkevich: (Online) Subgradient Methods for Structured Prediction, AIStats’07

Memory efficient
Fast and scalable
Structured Prediction for Programs

Program Analysis

30 nodes
400 edges

Time: milliseconds

MAP Inference

+ types
+ names

Prediction

Learning

Program Analysis

SSVM Learning

150 MB

Time: 10 Hours

names: 63%
Types: 81%

150 MB

alias analysis
call analysis
7M feature functions for names
70K feature functions for types
# 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>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>Trees</td>
<td></td>
</tr>
</tbody>
</table>

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

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

<table>
<thead>
<tr>
<th>Query Model</th>
<th>( \text{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>

[http://plml.ethz.ch/](http://plml.ethz.ch/)
Statistical Software Tools

**Write new code:**
Code Completion

```java
Camera camera = Camera.open();
camera.SetDisplayOrientation(90);
```

**Port code:**
Programming Language Translation

```c#
Console.WriteLine("Hi");
...```

**Understand code/security:**
JavaScript Deobfuscation
Type Prediction

**Debug code:**
Statistical Bug Detection

```python
... for x in range(a):
    print a[x]
```
**Statistical Software Tools**

**Write new code:**
Code Completion

```python
Camera camera = Camera.open();
camera.SetDisplayOrientation(90);
```

**Port code:**
Programming Language Translation

```csharp
Console.WriteLine("Hi");
...```

```java
System.out.println("Hi");
...
```

**Understand code/security:**
JavaScript Deobfuscation
Type Prediction

**Debug code:**
Statistical Bug Detection

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

likely error
Probabilistic Model for Code

Model is a key part of the Statistical Programming Tools

Goal: score programs
Select best among several candidates
Probabilistic Model for Code

Model is a key part of the Statistical Programming Tools

Goal: score programs
Select best among several candidates

Example: Which function is more likely?

```
function area(a) {
    return a.width * a.height
}
```

```
function area(a) {
    return a.width * a.close()
}
```
Model is a key part of Statistical Programming Tools

Example:

```javascript
function area(a) {
    return a.width * a.height;
}
```

Goal: score programs
Select best among several candidates
Probabilistic Model for Code

Directly applicable to code completion, but is a key statistical component for many others tasks: e.g. natural language to code, statistical bug localization
Model Requirements

Existing Programs

Learning

Model

Widely Applicable
Efficient Learning
High Precision
Explainable Predictions

Probabilistic Model
Observation

Regularities in code are similar to regularities in natural language

The quick brown fox jumps over the lazy ?
Observation

Regularities in code are similar to regularities in natural language

The quick brown fox jumps over the lazy dog
Observation

Regularities in code are similar to regularities in natural language

The quick brown fox jumps over the lazy dog

```python
file = open(filename, "r")
file.??
```
Observation

Regularities in code are similar to regularities in natural language

The quick brown fox jumps over the lazy dog

```python
file = open(filename, "r")
file.read()
```
N-gram Language Model

Conditional probability only on previous n-1 words

$$P(w_i | w_1 \ldots w_{i-1}) \approx P(w_i | w_{i-n+1} \ldots w_{i-1}) \approx \frac{\#(w_{i-n+1} \ldots w_{i-1} w_i)}{\#(w_{i-n+1} \ldots w_{i-1})}$$

$\#(n\text{-gram})$ - number of occurrences of n-gram in training data
N-gram Language Model

Conditional probability only on previous n-1 words

\[ P(w_i | w_1 ... w_{i-1}) \approx P(w_i | w_{i-n+1} ... w_{i-1}) \approx \frac{\#(w_{i-n+1} ... w_{i-1}w_i)}{\#(w_{i-n+1} ... w_{i-1})} \]

Training is achieved by counting n-grams (3-gram)

\[ P(\text{jumped} | \text{The quick brown fox}) \approx P(\text{jumped} | \text{brown fox}) \approx \frac{\#(\text{brown fox jumped})}{\#(\text{brown fox})} \]

\#(n-gram) - number of occurrences of n-gram in training data
## N-gram Language Model

### Training

```python
return f.height * scale;

f.open(mode);

f.close();

2 * f.width;

f.close()
```

### Prediction

```python
f.width + f.?
```
N-gram Language Model

Training (3-gram model)

return f. height * scale;

f. open (mode);

f. close ();

2 * f. width;

f. close ()

3-gram

Prediction

f. width + f. ?
Training (3-gram model)

```c
return f . height * scale;

f . open (mode);

f . close ();

2 * f . width;

f . close ()
```

Prediction

```
f . width + f . ? P
```

<table>
<thead>
<tr>
<th></th>
<th>P</th>
</tr>
</thead>
<tbody>
<tr>
<td>close</td>
<td>0.4</td>
</tr>
<tr>
<td>open</td>
<td>0.2</td>
</tr>
<tr>
<td>width</td>
<td>0.2</td>
</tr>
<tr>
<td>height</td>
<td>0.2</td>
</tr>
</tbody>
</table>
N-gram Language Model

<table>
<thead>
<tr>
<th>Conditioning</th>
<th>Accuracy</th>
</tr>
</thead>
<tbody>
<tr>
<td>Last two tokens, Hindle et. al. [ICSE’12]</td>
<td>22.2%</td>
</tr>
</tbody>
</table>

Main Problem: $f.\text{width} + f.? \mid P$

**Bad context** leads to bad probability estimates

<table>
<thead>
<tr>
<th></th>
<th>Probability</th>
</tr>
</thead>
<tbody>
<tr>
<td>close</td>
<td>0.4</td>
</tr>
<tr>
<td>open</td>
<td>0.2</td>
</tr>
<tr>
<td>width</td>
<td>0.2</td>
</tr>
<tr>
<td>height</td>
<td>0.2</td>
</tr>
</tbody>
</table>
Better Context

Training

\[ \text{return } x \cdot \text{width} \times x \cdot \text{height} \]

\[ \text{return } y \cdot \text{width} \times y \cdot \text{height} \]

\[ \text{area} = s \cdot \text{width} \times s \cdot \text{height} \]

\[ s \cdot \text{width} = s \cdot \text{width} + 10 \]

Q. depth \times q \cdot \text{width} \times q \cdot \text{height} 

Prediction

\[ f \cdot \text{width} + f \cdot \text{?} \]
Better Context

Training

\[
\text{return } x \cdot \text{width} \times x \cdot \text{height}
\]

\[
\text{return } y \cdot \text{width} \times y \cdot \text{height}
\]

\[
\text{area} = s \cdot \text{width} \times s \cdot \text{height}
\]

\[
s \cdot \text{width} = s \cdot \text{width} + 10
\]

\[
\text{q.depth} \times q \cdot \text{width} \times q \cdot \text{height}
\]

Prediction

\[
f \cdot \text{width} + f \cdot ?
\]

Context: relevant for this prediction

<table>
<thead>
<tr>
<th></th>
<th>height</th>
<th>width</th>
<th>open</th>
<th>close</th>
</tr>
</thead>
<tbody>
<tr>
<td>q.depth</td>
<td>0.8</td>
<td>0.2</td>
<td>0.0</td>
<td>0.0</td>
</tr>
</tbody>
</table>
### Better Context

<table>
<thead>
<tr>
<th>Conditioning</th>
<th>Accuracy</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>
</tbody>
</table>
### JavaScript APIs

<table>
<thead>
<tr>
<th>Conditioning</th>
<th>Accuracy</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>
</tbody>
</table>

*is this the best we can do?*
## JavaScript APIs

<table>
<thead>
<tr>
<th>Conditioning</th>
<th>Accuracy</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>
</tbody>
</table>

Is this the best we can do?

### Program synthesis

66.4%
<table>
<thead>
<tr>
<th>Conditioning</th>
<th>Accuracy</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>Last three APIs</td>
<td></td>
</tr>
<tr>
<td>Declaration Site + Last two APIs</td>
<td></td>
</tr>
<tr>
<td>Variable Name + Method Name + Last API</td>
<td></td>
</tr>
</tbody>
</table>

How do we know that which is the best context?
<table>
<thead>
<tr>
<th>Conditioning</th>
<th>Accuracy</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>Last three APIs</td>
<td></td>
</tr>
<tr>
<td>Declaration Site + Last two APIs</td>
<td></td>
</tr>
<tr>
<td>Variable Name + Method Name + Last API</td>
<td></td>
</tr>
</tbody>
</table>

How do we know that which is the best context?
JavaScript APIs

Identifiers
Strings
Numbers
Arguments
Properties
Statements
RegExp
Structure
Solution: Synthesise the Best Model

{\text{width}} = \textcolor{blue}{f} ( x.\text{width} + x.\textcolor{orange}{?} )

Synthesise a function $f$ from a domain specific language that explains the data
Function Examples

\[ f(p_1) = \{ \]
Function Examples

\[ f(p_1) = \begin{cases} 
\text{for,} & \text{if,} \\
\text{length} == 0 & \text{?}
\end{cases} \]
Function Examples

\[ f(p_1) = \begin{cases} 
  \text{for,} & \text{if,} \\
  \text{length} = \emptyset 
\end{cases} \]

\[ f(p_2) = \begin{cases} 
  \text{if,} 
\end{cases} \]

for \( j = 0; j < \text{groups.length}; j++ \) {
  \text{idsInGroup} = \text{groups}[j].filter(
    \text{function(id) { return id >= 42; }}
  );
  \text{if}(\text{idsInGroup.length} == 0) {
    \text{elem.notify(..., { position: 'top', hide: false, ... })};
  }
}


Function Examples

\[ f(p_1) = \begin{cases} \text{for,} \\ \text{if,} \\ \text{length}=0 \end{cases} \]

\[ f(p_2) = \begin{cases} \text{notify,} \\ \text{position,} \\ \text{hide} \end{cases} \]

```javascript
for (j = 0; j < groups.length; j++) {
    idsInGroup = groups[j].filter(
        function(id) { return id >= 42; }
    );
    if (idsInGroup.length == 0) {
        ?
    }
}

elem.notify(..., {
    position: 'top',
    hide: false,
    ?
});
```
Synthesise a function $f$ from a domain specific language that explains the data.
Overview

Synthesise a function $f$ from a domain specific language that explains the data.
Function Representation

In general:
Unrestricted programs (Turing complete)

Our Work:
TCond Language for navigating over trees and accumulating context

TCond ::= ε | WriteOp TCond | MoveOp TCond | BranchProg

BranchProg ::= if pred(x) then TCond else TCond

MoveOp ::= Up, Left, Right, DownFirst, DownLast, NextDFS, PrevDFS, NextLeaf, PrevLeaf, PrevNodeType, PrevNodeValue, PrevNodeContext

WriteOp ::= WriteValue, WriteType, WritePos
Expressing functions: TCond Language

TCond ::= ε | WriteOp TCond | MoveOp TCond | BranchProg

BranchProg ::= if pred(x) then TCond else TCond

MoveOp ::= Up, Left, Right, DownFirst, DownLast, NextDFS, PrevDFS, NextLeaf, PrevLeaf, PrevNodeType, PrevNodeValue, PrevNodeContext

WriteOp ::= WriteValue, WriteType, WritePos

γ ← γ · □
elem.notify(
    ...
    ...
    {
        position: 'top',
        hide: false,
        ?
    }
);
elem.notify(
    ...,
    ...
    {
        position: 'top',
        hide: false,
    }
);
Example

Query

elem.notify(
  ⋯,
  ⋯,
  {
    position: 'top',
    hide: false,
    ⋯
  }
);

TCond

Left
WriteValue
Up
WritePos

γ

{ }
{hide}
{hide}
{hide, 3}
Example

Query

elem.notify(
  ... ,
  ...

  {
    position: 'top',
    hide: false,
    ?
  }

);
Example

Query

elem.notify(
  ..., 
  ..., 
  {
    position: 'top',
    hide: false,
    ...
  }
);

TCond

  Left
  WriteValue
  Up
  WritePos
  Up
  DownFirst
  DownLast
  WriteValue

  γ

  {} 
  {hide} 
  {hide} 
  {hide, 3} 
  {hide, 3} 
  {hide, 3} 
  {hide, 3} 
  {hide, 3, notify}

{ Previous Property, Parameter Position, API name }
Results

Probabilistic Model of JavaScript Language

20k Learning  100k Training  50k Blind Set

GitHub
# JavaScript APIs

<table>
<thead>
<tr>
<th>Conditioning</th>
<th>Accuracy</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>Program synthesis</strong></td>
<td>66.4%</td>
</tr>
</tbody>
</table>
## JavaScript Structure

<table>
<thead>
<tr>
<th>Model</th>
<th>Accuracy</th>
</tr>
</thead>
<tbody>
<tr>
<td>PCFG</td>
<td>51.1%</td>
</tr>
<tr>
<td>N-gram</td>
<td>71.3%</td>
</tr>
<tr>
<td>Naïve Bayes</td>
<td>44.2%</td>
</tr>
<tr>
<td>SVM</td>
<td>70.5%</td>
</tr>
<tr>
<td>Program synthesis</td>
<td>81.5%</td>
</tr>
</tbody>
</table>
# JavaScript Values

<table>
<thead>
<tr>
<th>Accuracy</th>
<th>Example</th>
</tr>
</thead>
<tbody>
<tr>
<td>Identifier</td>
<td>62%</td>
</tr>
<tr>
<td>Property</td>
<td>65%</td>
</tr>
<tr>
<td>String</td>
<td>52%</td>
</tr>
<tr>
<td>Number</td>
<td>64%</td>
</tr>
<tr>
<td>RegExp</td>
<td>66%</td>
</tr>
<tr>
<td>UnaryExpr</td>
<td>97%</td>
</tr>
<tr>
<td>BinaryExpr</td>
<td>74%</td>
</tr>
<tr>
<td>LogicalExpr</td>
<td>92%</td>
</tr>
</tbody>
</table>
Model Requirements

Existing Programs

Learning

Model

- Widely Applicable
- Efficient Learning
- High Precision
- Explainable Predictions

Probabilistic Model
Learning

TCond ::= ε | WriteOp TCond | MoveOp TCond
MoveOp ::= Up, Left, Right, ...
WriteOp ::= WriteValue, WriteType, ...

Program Synthesis
- Enumerative search
- Genetic programming
- Decision tree learning
- MCMC

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

Dataset

\[ |d| \ll |D| \]
\[ |\text{cost}(d, f) - \text{cost}(D, f)| < \varepsilon \]

Representative sampling
Applications

- Writing new Code
- Porting Code
- Finding Bugs
- Allocation site Analysis
- Points-to Analysis
- Handwriting Recognition
- Speech Recognition
- Proobabilitic Model
- Static Analysis
- Neural Networks
- Character-level Language Models

The quick brown fox jumps over the lazy dog
Work @ ETH Zurich
Learning from “Big Code”

Key Idea:
Learn a function $f$ that explains the data. The function dynamically obtains the best conditioning context for a given query.

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

Name and Types
Prediction
...

Intermediate Representation
Graphical Models
Feature Vectors

Analyze Program (PL)
Structured SVM

Train Model (ML)
Greedy MAP Inference

Query Model

http://plml.ethz.ch/

Applications

NICE 2 Predict