Detecting Variable Misuses with Graph Neural Networks

Miltos Allamanis
Joint work with Marc Brockschmidt and Mahmoud Khademi

https://arxiv.org/abs/1711.00740
Motivating Example

```csharp
var clazz = classTypes["Root"].Single() as JsonCodeGenerator.ClassType;
Assert.NotNull(clazz);

var first = classTypes["RecClass"].Single() as JsonCodeGenerator.ClassType;
Assert.NotNull(first);

Assert.Equal("string", first.Properties["Name"].Name);
Assert.False(clazz.Properties["Name"].IsArray);
```

Possible type-correct options: clazz, first
Graph Neural Networks

A brief introduction

Localized vs Distributed (Vector) Representations

\[ P_D(\pi | f(c)) \]

\[ \mathbb{R}^D \]
Gated Graph Neural Networks

![Graph Representation of Problem](image1)

![Initial Representation of each node](image2)

Graph Neural Networks: Message Propagation

Graph Neural Networks: Unrolling

Graph Neural Networks: Unrolling

- node selection
- node classification
- graph classification

Detecting Variable Misuses

with Graph Neural Networks

https://arxiv.org/abs/1711.00740
Representing Program Structure as a Graph
Representing Program Structure as a Graph

\((x^1, y^2) = \text{Foo}();\)

\(\text{while } (x^3 > 0) \ x^4 = x^5 + y^6\)
Representing Program Structure as a Graph

Additional Edge Types:
- LastLexicalUse
- ReturnsTo
- FormalArgName

```c
int foo(int sum) {
    ...
    return x;
}
```
Representing Program Structure as a Graph

Additional Edge Types:
• LastLexicalUse
• ReturnsTo
• FormalArgName

foo(result);

sum
Graph Representation for Variable Misuse

```csharp
var clazz = classTypes["Root"].Single() as JsonCodeGenerator.ClassType;
Assert.NotNull(clazz);

var first = classTypes["RecClass"].Single() as JsonCodeGenerator.ClassType;
Assert.NotNull(first);

Assert.Equal("string", first.Properties["Name"].Name);
Assert.False(clazz.Properties["Name"].IsArray);
```

Possible type-correct options: clazz, first
Graph Representation for Variable Misuse

var clazz = classTypes["Root"].Single() as JsonCodeGenerator.ClassType;
Assert.NotNull(clazz);

var first = classTypes["RecClass"].Single() as JsonCodeGenerator.ClassType;
Assert.NotNull(SLOT); first clazz

Assert.Equal("string", first.Properties["Name"].Name);
Assert.False(clazz.Properties["Name"].IsArray);

Goal: make the representation of SLOT as close as possible to the representation of the correct candidate node
Representing Variable Type Information

\[ \tau^*(v) = \{ \tau_{\text{List}<\text{string}>}, \tau_{\text{IList}}, \tau_{\text{object}}, \ldots \} \]

- \( r_{\text{List}<\text{string}>} \)
- \( r_{\text{IList}} \)
- \( r_{\text{object}} \)

Elementwise Max

\[ r_{\tau^*(v)} \]
Representing Nodes

classTypes → class, types

$\mathbf{Avg}$

$\mathbf{r}_{\tau^*(v)}$
Implementation

• Sparse TensorFlow implementation
• 16 edge types (forward + backward)
• ~900 nodes/graph  ~8k edges/graph

Stats
• 55 graphs/s during training
• 219 graphs/s during testing
<table>
<thead>
<tr>
<th>Name</th>
<th>Git SHA</th>
<th>kLOCs</th>
<th>Slots</th>
<th>Vars</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Akka.NET</td>
<td>719335a1</td>
<td>240</td>
<td>51.3k</td>
<td>51.2k</td>
<td>Actor-based Concurrent &amp; Distributed Framework</td>
</tr>
<tr>
<td>AutoMapper</td>
<td>2ca7c2b5</td>
<td>46</td>
<td>3.7k</td>
<td>10.7k</td>
<td>Object-to-Object Mapping Library</td>
</tr>
<tr>
<td>BenchmarkDotNet</td>
<td>1670ca34</td>
<td>28</td>
<td>5.1k</td>
<td>6.1k</td>
<td>Benchmarking Library</td>
</tr>
<tr>
<td>Bot.Builder</td>
<td>190117c3</td>
<td>44</td>
<td>6.4k</td>
<td>8.7k</td>
<td>SDK for Building Bots</td>
</tr>
<tr>
<td>choco</td>
<td>93985688</td>
<td>36</td>
<td>3.8k</td>
<td>5.2k</td>
<td>Windows Package Manager</td>
</tr>
<tr>
<td>commandline†</td>
<td>09677b16</td>
<td>11</td>
<td>1.1k</td>
<td>2.3k</td>
<td>Command Line Parser</td>
</tr>
<tr>
<td>CommonMark.NET Dev</td>
<td>f3d54530</td>
<td>14</td>
<td>2.6k</td>
<td>1.4k</td>
<td>Markdown Parser</td>
</tr>
<tr>
<td>Dapper</td>
<td>931c700d</td>
<td>18</td>
<td>3.3k</td>
<td>4.7k</td>
<td>Object Mapper Library</td>
</tr>
<tr>
<td>EntityFramework</td>
<td>fa0b7ec8</td>
<td>263</td>
<td>33.4k</td>
<td>39.3k</td>
<td>Object-Relational Mapper</td>
</tr>
<tr>
<td>Hangfire</td>
<td>ffc4912f</td>
<td>33</td>
<td>3.6k</td>
<td>6.1k</td>
<td>Background Job Processing Library</td>
</tr>
<tr>
<td>Humanizer†</td>
<td>cc11a77e</td>
<td>27</td>
<td>2.4k</td>
<td>4.4k</td>
<td>String Manipulation and Formatting</td>
</tr>
<tr>
<td>Lean†</td>
<td>f574bdf8d</td>
<td>190</td>
<td>26.4k</td>
<td>28.3k</td>
<td>Algorithmic Trading Engine</td>
</tr>
<tr>
<td>Nancy</td>
<td>72e1f614</td>
<td>70</td>
<td>7.5k</td>
<td>15.7k</td>
<td>HTTP Service Framework</td>
</tr>
<tr>
<td>Newtonsoft.Json</td>
<td>6057d9b8</td>
<td>123</td>
<td>14.9k</td>
<td>16.1k</td>
<td>JSON Library</td>
</tr>
<tr>
<td>Ninject</td>
<td>70062976</td>
<td>13</td>
<td>0.7k</td>
<td>2.1k</td>
<td>Code Injection Library</td>
</tr>
<tr>
<td>NLog</td>
<td>643e326a</td>
<td>75</td>
<td>8.3k</td>
<td>11.0k</td>
<td>Logging Library</td>
</tr>
<tr>
<td>Opserver</td>
<td>51b032e7</td>
<td>24</td>
<td>3.7k</td>
<td>4.5k</td>
<td>Monitoring System</td>
</tr>
<tr>
<td>OptiKey</td>
<td>7d35c718</td>
<td>34</td>
<td>6.1k</td>
<td>3.9k</td>
<td>Assistive On-Screen Keyboard</td>
</tr>
<tr>
<td>Orleans</td>
<td>e0d6a150</td>
<td>300</td>
<td>30.7k</td>
<td>35.6k</td>
<td>Distributed Virtual Actor Model</td>
</tr>
<tr>
<td>Polly</td>
<td>0afdb7c2</td>
<td>32</td>
<td>3.8k</td>
<td>9.1k</td>
<td>Resilence &amp; Transient Fault Handling Library</td>
</tr>
<tr>
<td>quartznet</td>
<td>b33e6f86</td>
<td>49</td>
<td>9.6k</td>
<td>9.8k</td>
<td>Scheduler</td>
</tr>
<tr>
<td>RavenDB Dev</td>
<td>55230922</td>
<td>647</td>
<td>78.0k</td>
<td>82.7k</td>
<td>Document Database</td>
</tr>
<tr>
<td>RestSharp</td>
<td>70de357b</td>
<td>20</td>
<td>4.0k</td>
<td>4.5k</td>
<td>REST and HTTP API Client Library</td>
</tr>
<tr>
<td>Rx.NET</td>
<td>2d146fe5</td>
<td>180</td>
<td>14.0k</td>
<td>21.9k</td>
<td>Reactive Language Extensions</td>
</tr>
<tr>
<td>scriptcs</td>
<td>f3cc8bc8</td>
<td>18</td>
<td>2.7k</td>
<td>4.3k</td>
<td>C# Text Editor</td>
</tr>
<tr>
<td>ServiceStack</td>
<td>6d59da75</td>
<td>231</td>
<td>38.0k</td>
<td>46.2k</td>
<td>Web Framework</td>
</tr>
<tr>
<td>ShareX</td>
<td>718d711</td>
<td>125</td>
<td>22.3k</td>
<td>18.1k</td>
<td>Sharing Application</td>
</tr>
<tr>
<td>SignalR</td>
<td>fa88089e</td>
<td>53</td>
<td>6.5k</td>
<td>10.5k</td>
<td>Push Notification Framework</td>
</tr>
<tr>
<td>Wox</td>
<td>cda6272</td>
<td>13</td>
<td>2.0k</td>
<td>2.1k</td>
<td>Application Launcher</td>
</tr>
</tbody>
</table>
# Quantitative Results

<table>
<thead>
<tr>
<th>Accuracy (%)</th>
<th>Local Model</th>
<th>Avg BiRNN</th>
<th>GGNN</th>
</tr>
</thead>
<tbody>
<tr>
<td>Seen Projects</td>
<td>15.8</td>
<td>73.5</td>
<td>82.1</td>
</tr>
<tr>
<td>Unseen Projects</td>
<td>13.8</td>
<td>59.7</td>
<td>68.6</td>
</tr>
</tbody>
</table>

3.8 type-correct alternative variables per slot (median 3, σ = 2.6)
```csharp
bool TryFindGlobalDirectivesFile(string baseDirectory, string fullPath, out string path) {
    baseDirectory = baseDirectory.TrimEnd(Path.DirectorySeparatorChar);
    var directivesDirectory = Path.GetDirectoryName(baseDirectory) .TrimEnd(Path.DirectorySeparatorChar);
    while (directivesDirectory != null && directivesDirectory.Length >= baseDirectory.Length) {
        path = Path.Combine(directivesDirectory, GlobalDirectivesFileName);
        if (File.Exists(path)) return true;
        directivesDirectory = Path.GetDirectoryName(directivesDirectory) .TrimEnd(Path.DirectorySeparatorChar);
    }
    path = null;
    return false;
}
```
```csharp
bool TryFindGlobalDirectivesFile(string baseDirectory, string fullPath, out string path) {
    baseDirectory = baseDirectory.TrimEnd(Path.DirectorySeparatorChar);
    var directivesDirectory = Path.GetDirectoryName(fullPath) .TrimEnd(Path.DirectorySeparatorChar);
    while (directivesDirectory != null && directivesDirectory.Length >= baseDirectory.Length) {
        path = Path.Combine(directivesDirectory, GlobalDirectivesFileFileName);
        if (File.Exists(path)) return true;

        directivesDirectory = Path.GetDirectoryName(directivesDirectory) .TrimEnd(Path.DirectorySeparatorChar);
    }
    path = null;
    return false;
}
```

What the model sees...
Qualitative Results

```csharp
public void MergeFrom(pb::CodedInputStream input) {
    uint tag;
    while ((tag = input.ReadTag()) != 0) {
        switch (tag) {
            default: {
                input.SkipLastField();
                break;
            }
            case 10: {
                .AddEntriesFrom(input, _repeated_payload_codec);
                break;
            }
        }
    }
}
```

#1 Payload: 66%, payload_: 44%
public override bool IsDisposed
{
    get
    {
        lock (  )
        {
            return ;
        }
    }
}
Qualitative Results

```javascript
var response = ResultsFilter(typeof(TResponse), #1, #2, request);
```

#1 httpMethod: 99%, absoluteUrl: 1%, UserName: 0%, UserAgent: 0%

#2 absoluteUrl: 99%, httpMethod: 1%, UserName: 0%, UserAgent: 0%
private bool BecomingCommand(object message)
{
    if (ReceiveCommand(#1) return true;
    if (#2.ToString() == #3) #4.Tell(#5);
    else return false;
    return true;
}
Detecting Real-Life Bugs

```csharp
public ArraySegment<byte> ReadBytes(int length){
    int size = Math.Min(length, _len - _pos);
    var buffer = EnsureTempBuffer(length);
    var used = Read(buffer, 0, size);
    return new ArraySegment<byte>(buffer, 0, used);
}
```
protected void ValidateRestorePreconditions(string backupFilename) {
    if (IsValidBackup(backupFilename) == false) {
        output("Error:" + backupLocation + " doesn’t look like a valid backup");
        output("Error: Restore Canceled");
        throw new InvalidOperationException(
            backupLocation + " doesn’t look like a valid backup");
    }
    ...
}
Closing Thoughts

Exploit Rich Structure of Code
Graph Neural Networks Scale

A survey: https://ml4code.github.io [Please, contribute]