Towards Learned Program Analyses with Machine Learning

Miltos Allamanis
Microsoft Research Cambridge

Joint work with Earl T. Barr, Marc Brockschmidt, Santanu Dash, Mahmoud Khademi
Deep Program Understanding

- **Deep Learning**
  - ✓ Understands images/language/speech
  - ✓ Finds patterns in noisy data
  - Requires many samples
  - Handling structured data is hard

- **DPU**
  - ✓ Interpretable
  - ✓ Generalisation verifiable
  - Manual effort
  - Limited to specialists

- **Program Structure**
Source Code and Natural Language
Code as...

- Data → Software Engineering (SE) Tools
- Machine Learning (ML) component → Artificial Intelligence (AI) Tool
Research in ML4Code

- Infer latent intent
- Ambiguous information

https://ml4code.github.io

A Survey of Machine Learning for Big Code and Naturalness

MILTIADIS ALLAMANIS, Microsoft Research
EARL T. BARR, University College London
PREMKUMAR DEVANBU, University of California, Davis
CHARLES SUTTON, University of Edinburgh and The Alan Turing Institute

Research at the intersection of machine learning, programming languages, and software engineering has recently taken important steps in proposing learnable probabilistic models of source code that exploit code’s abundance of patterns. In this article, we survey this work. We contrast programming languages against natural languages and discuss how these similarities and differences drive the design of probabilistic models. We present a taxonomy based on the underlying design principles of each model and use it to navigate the literature. Then, we review how researchers have adapted these models to application areas and discuss crosscutting and application-specific challenges and opportunities.
Highlights: Program Analysis with Machine Learning
Code Autocompletion

Text text = new Text(parent, SWT.NONE);

http://www.eclipse.org/recommenders/

https://visualstudio.microsoft.com/services/intellicode/
public class TextRunnerTest extends TestCase {
    void execTest(String testClass, boolean success) throws Exception {
        ...
        InputStream input = p.getInputStream();
        while ((input.read()) != -1);
        ...
    }
    ...
}

Suggested Name

input (81.9%)
Argument Swapping

setSize(width, height) or setSize(height, width)

promise.done(res, err) or promise.done(err, res)
Predicting Types

Deep Learning Type Inference
V. Hellendoorn, C. Bird, E.T. Barr, M. Allamanis. 2018

Predicting Program Properties from Code
V. Raychev, M. Vechev, A. Krause. 2015

http://jsnice.org/
- Variable Misuse
- Learning Nominal Type Refinements
Variable Misuse

```csharp
// Create or update the document.
var newDocument = await cosmosClient.UpsertDocumentAsync(cosmosDbCollectionUri, document);

if (updateRecord)
{
    logger.WriteLog($"Updated {existingDocument} to {newDocument}"");
}
else
{
    logger.WriteLog($"Added {existingDocument}";

Based on this repo’s code patterns, did you intend to use ‘newDocument’ (confidence 92%) rather than ‘existingDocument’ (confidence 7%) here? Review is recommended by Research bot’s Variable Misuse analysis.

Allamanis et al. “Learning to Represent Programs with Graphs”. 2018
Inferring Type Refinements

Conceptual Types

“a password”

Defined Types

\[
\text{string} \quad \text{password;}
\]

“a JSON string”

\[
\text{string} \quad \text{data} = \text{Json.Load();}
\]

Latent; we don’t observe the conceptual types.

Defined explicitly by the programmer.

Dash et al. 2018 “RefiNym: Using Names to Refine Types”
Variable Misuse

with Graph Neural Networks
Programs as Graphs

```c
int SumPositive(int[] arr, int lim) {
    int sum = 0;
    for (int i = 0; i < lim; i++)
        if (arr[i] > 0)
            sum += arr[i];

    return sum;
}
```
Programs as Graphs: Syntax

```
Assert.NotNull(clazz);
```

- Next Token
- AST Child
Programs as Graphs: Data Flow

\[(x, y) = \text{Foo}();\]

\textbf{while} \ (x > 0) \]

\[x = x + y;\]
int SumPositive(int[] arr, int lim) {
    int sum = 0;
    for (int i = 0; i < lim; ++i)
        if (arr[i] > 0)
            += arr[i];
    return sum;
}

~900 nodes/graph  ~8k edges/graph
Graph Neural Networks

Li et al (2015). Gated Graph Sequence Neural Networks.

Neural Message Passing

1. Current Neighbor States
   - D
   - E

2. Prepare “Message”
   - $\text{msg} = f(D \rightarrow F)$
   - $\text{msg} = f(E \rightarrow F)$

3. Summarize Received Information
   - $h_{t-1}^n$

4. Next Node State
   - $h_{t}^n$
Gated GNNs


\[ x = \sum_{n' \in \text{neig}(n)} E_{\tau(n' \rightarrow n)} h_{t-1}^{n'} \]
Graph Neural Networks: Message Passing
GNNs: Synchronous Message Passing (All-to-All)
Graph Neural Networks: Output

- node selection
- node classification
- graph classification

Li et al (2015). Gated Graph Sequence Neural Networks.

https://github.com/Microsoft/gated-graph-neural-network-samples
Initial Node Representations

Label: out_file_prefix
Type: string

out, file, prefix → Embed → Average → Concat

string, object, ... → Embed → Max Pool
Variable Misuse Task

```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

⚠️ Not easy to catch with static analysis tools.
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

```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);
```

**Goal:** make the representation of SLOT as close as possible to the representation of the correct candidate node

\[
f(h_T^{SLOT}, h_T^{first}) \gg f(h_T^{SLOT}, h_T^{clazz})
\]
<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>BotBuilder</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>9398568</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.NETDev</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>f574bfd7</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>15.7</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>7006297f</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>0afdbc32</td>
<td>32</td>
<td>3.8k</td>
<td>9.1k</td>
<td>Resilience &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>rvendbDev</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>f3cc8bc6</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>718dd711</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>cdaf6722</td>
<td>13</td>
<td>2.0k</td>
<td>2.1k</td>
<td>Application Launcher</td>
</tr>
</tbody>
</table>
Quantitative Results – Variable Misuse

<table>
<thead>
<tr>
<th></th>
<th>BiGRU</th>
<th>BiGRU+Dataflow</th>
<th>GGNN</th>
</tr>
</thead>
<tbody>
<tr>
<td>Seen Projects</td>
<td>50.0</td>
<td>73.7</td>
<td>85.5</td>
</tr>
</tbody>
</table>

Seen Projects: 24 F/OSS C# projects (2060 kLOC): Used for train and test
3.8 type-correct alternative variables per slot (median 3, \( \sigma = 2.6 \))
# Quantitative Results – Variable Misuse

<table>
<thead>
<tr>
<th></th>
<th>Accuracy (%)</th>
<th>BiGRU</th>
<th>BiGRU+Dataflow</th>
<th>GGNN</th>
</tr>
</thead>
<tbody>
<tr>
<td>Seen Projects</td>
<td>50.0</td>
<td>73.7</td>
<td>85.5</td>
<td></td>
</tr>
<tr>
<td>Unseen Projects</td>
<td>28.9</td>
<td>60.2</td>
<td>78.2</td>
<td></td>
</tr>
</tbody>
</table>

Seen Projects: 24 F/OSS C# projects (2060 kLOC): Used for train and test
Unseen Projects: 3 F/OSS C# projects (228 kLOC): Used only for test
3.8 type-correct alternative variables per slot (median 3, σ= 2.6)
// Create or update the document.
var newDocument = await cosmosClient.UpsertDocumentAsync(cosmosDbCollectionUri, document);

if (updateRecord)
{
    logger.WriteLog("Updated {existingDocument} to {newDocument}");
}
else
{
    logger.WriteLog("Added {existingDocument}");
}

Based on this repo's code patterns, did you intend to use 'newDocument' (confidence 92%) rather than 'existingDocument' (confidence 7%) here? Review is recommended by Research bot's Variable Misuse analysis.

+1
Refining Types

...using natural language.
def addToCart(cartId: String, productId: String, providerId: String)

    addToCart(productId, providerId, cartId)

username := password

temperature + numOfOranges
Conceptual Types

"a password" → `string password;`

"a JSON string" → `string data = Json.Load();`

Latent; we don’t observe the conceptual types.

Defined explicitly by the programmer.
string EncryptAndSignCookie(string cookieValue, FormsAuthenticationConfiguration config) {
    string encryptedCookie =
        config.CryptographyConfiguration.EncryptionProvider.Encrypt(cookieValue);
    var hmacBytes = GenerateHmac(encryptedCookie, config);
    string hmacString = Convert.ToBase64String(hmacBytes);
    return hmacString + encryptedCookie;
}
Name Flow Graphs

**Representation:**
- [Static] Data Flow
- Identifier Names
Constructing Name Flows – Assignment

```
filePath

fileDir = filePath

"../app.exe"

execDir = "../app.exe"
```
Constructing Name Flows – Returns

def NormalizePath(...)
    ...
    return normalized;
}
Constructing Name Flows – Function Calls

```
NormalizePath

normedPath = NormalizePath(...)  
normedPath
```
Constructing Name Flows – Actuals to Formals

```python
def Exists(pathToFile)
    path
```

pathToFile

path

pathToFile

path
Constructing Name Flows – Override

```python
def override string Foo(string bar) { ... }

def string Foo(string arg) { ... }
```
Constructing Name Flows – Summary

- Capture type-correct flows
- Capture names of variables/methods
From Dataflow to Nominal Type Refinements

Dataflow of string variables/method and their names.

Nominal string refinements.

through information theory...
We want both

\[ P(\text{cluster}|\text{name}) \]

\[ P(\text{name}|\text{cluster}) \]

to have low entropy
Variation of Information

\[ VI(N, C) = H(N|C) + H(C|N) = H(C, N) - I(C, N) \]
Variation of Information: Objective

\[ C^* = \arg\min_C \text{VI}(N, C) \text{ s.t. } \exists R = (C, \subseteq) \]
Variation of Information: Intuition

\[ VI(N, \{c_1\}) > VI(N, \{c_2, c_3\}) \]
Variation of Information: Intuition

\[ VI(N, \{c_1\}) < VI(N, \{c_2, c_3\}) \]
Modeling Names

\[ Vl(N, C) = H(N|C) + H(C|N) \]
\[ = H(N, C) - I(N, C) \]

\[ H(N|C) = - \sum_{c \in C} P(c) \sum_{n \in N} P(n|c) \log P(n|c) \]
Modeling Names – Subtokens

\[ P_{emp}(\text{"dir"}|c) = \frac{2}{6} \]

- Define a multinomial probability over subtokens.
- Add a prior for smoothing:

\[
P(s|c) = \frac{|s \in S_c| + a P_{all}(s)}{|S_c| + a}
\]
From Dataflow to Nominal Type Refinements

\[ C^* = \text{argmin}_C V I(N, C) \text{ s.t. } \exists R = (C, \subseteq) \]
Greedy Optimization of VI – Split
Greedy Optimization of VI – Split
Greedy Optimization of VI – Partial Order Point
Greedy Optimization of VI – Merge
Greedy Optimization of VI – Merge
<p>| | | | | | | | | | | |</p>
<table>
<thead>
<tr>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>path, Path, originalRequestPath, modifiedRequestPath, owinRequestPath, &quot;/&quot;</td>
<td>emailConstraint/&quot;, contentPath, basePath, IViewEngineHost::ExpandPath, AspNetRootPathProvider::GetRootPath, &quot;/&quot;, DiagnosticsConfiguration::GetNormalizedPath, NancyContext::ToFullPath, ModulePath</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>2</td>
<td>DefaultCulture, defaultCulture, cookieCulture, cultureLetters, name</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>3</td>
<td>earlyExitReason, &quot;Requires Any Claim&quot;, &quot;Requires Claims&quot;, &quot;Requires Authentication&quot;, &quot;Accept&quot;</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>
| 4 | IObjectSerializer::Serialize, DefaultObjectSerializer::Serialize, JsonObject::ToString, 
SimpleJson::SerializeObject, HttpRequest::Collection::ToString |
<p>| 5 | method, Method, &quot;PUT&quot;, &quot;POST&quot;, &quot;PATCH&quot;, &quot;OPTIONS&quot;, &quot;HEAD&quot;, &quot;GET&quot;, &quot;DELETE&quot; |
| 6 | value, cookieValue, sourceString, &quot;SomeValue&quot;, cookieValueEncrypted, attemptedValue, decryptedValue, defaultValue |
| 7 | password, &quot;password&quot;, realPassword, plainText, Password |
| 8 | HttpRequest::UrlDecode, HttpRequest::UrlPathEncode, HttpRequest::UrlEncodeUnicode, redirectUrl, fallbackRedirectUrl, url, path |</p>
<table>
<thead>
<tr>
<th>Full name of node or constant value in bepuphysics</th>
</tr>
</thead>
<tbody>
<tr>
<td>damping, SuspensionDamping, starchDamping, dampingConstant, angularDamping, LinearDamping</td>
</tr>
<tr>
<td>currentDistance, distance3, candidateDistance, pointDistance, distanceFromMaximum, grabDistance, VariableLinearSpeedCurve::GetDistance, tempDistance</td>
</tr>
<tr>
<td>goalVelocity, driveSpeed, GoalSpeed</td>
</tr>
<tr>
<td>minRadius, MinimumRadius, Radius, minimumRadiusA, WrappedShape::ComputeMinimumRadius, topRadius, MaximumRadius, graphicalRadius, TransformableShape::ComputeMaximumRadius</td>
</tr>
<tr>
<td>blendedCoefficient, KineticFriction, dynamicCoefficient, KineticBreakingFrictionCoefficient</td>
</tr>
<tr>
<td>angle, myMaximumAngle, MinimumAngle, currentAngle, MaximumAngle, steeringAngle, MathHelper::WrapAngle</td>
</tr>
<tr>
<td>targetHeight, Height, ProneHeight, crouchingHeight, standingHeight</td>
</tr>
<tr>
<td>Mass, effectiveMass, newMassA, newMass</td>
</tr>
<tr>
<td>M22, m11, M44, resultM44, M43, intermediate, m31, X, Y, Z</td>
</tr>
</tbody>
</table>
Interplay between types and names.

Joint work with S. Dash, E.T. Barr (UCL)
UI/UX

ML Capabilities

Metrics

Low resources
• Given dataset \( \{(x_1, y_1), \ldots, (x_N, y_N)\} \)
• Minimize Loss \( \mathcal{L}(\theta) = \frac{1}{N} \sum_i L(f_\theta(x_i), y_i) \)
How to Tell a Compiler What We Think We Know?

Guy L. Steele Jr.
Software Architect, Oracle Labs

SPLASH Keynote
Friday, November 4, 2016

This Talk Is an Essay (I Didn’t Know Where It Would Go)

I started out wanting to tell things to a compiler (or IDE).
- Specifically, I want to tell a compiler far more than types.
- I thought the conclusion would be that compilers need theorem provers.

That’s not a bad goal. But I have ended up wanting much more:
- I want a conversational partner that will track what I am doing.
- I want it to react to context and intention.
- I want it to give me relevant information.

This is much harder than “Just the facts, Ma’am.”
Towards Learned Program Analyses with Machine Learning

Reasoning over Rich Structures

Learning from Human Aspects of Code

miltos1

https://miltos.allamanis.com