Machine Learning for Program Analysis

Miltos Allamanis

Microsoft Research, Cambridge

@miltos1

miltos.allamanis.com

Joint work with Marc Brockschmidt, Mahmoud Khademi, Hamel Husain, Ho-Hsiang Wu, Tiferet Gazit, Santanu Dash, Earl T. Barr
All Data AI — 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 is meant to be read by humans too.
Code Autocompletion

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

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

https://visualstudio.microsoft.com/services/intellicode/
Argument Swapping

**Declaration:** void foo(Duration responseTTLDuration, Duration frequencyCapDuration, List<A> slotResponse)

**Invocation:** foo(frequencyCapDuration, responseTTLDuration, slotResponse)

<table>
<thead>
<tr>
<th>Type</th>
<th>Parameter</th>
<th>Original argument</th>
<th>Correct argument</th>
</tr>
</thead>
<tbody>
<tr>
<td>Duration</td>
<td>responseTTLDuration</td>
<td>frequencyCapDuration</td>
<td>responseTTLDuration</td>
</tr>
<tr>
<td>Duration</td>
<td>frequencyCapDuration</td>
<td>responseTTLDuration</td>
<td>frequencyCapDuration</td>
</tr>
<tr>
<td>List&lt;A&gt;</td>
<td>slotResponse</td>
<td>slotResponse</td>
<td>slotResponse</td>
</tr>
</tbody>
</table>

Rice et al. 2017 “Detecting Argument Selection Defects”
Research in ML+Code

• 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 cross-cutting and application-specific challenges and opportunities.

CCS Concepts: Computing methodologies → Machine learning; Natural language processing; Software and its engineering → Software analysis; Software development; Synthetic programming
- Understanding Code
- Types & Machine Learning
- What’s next?
“Understanding” Source Code

...with graph neural networks.
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.
A Program Graph Representation

```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;
}
```
Assert.NotNull(clazz);
A Program Graph Representation: Data Flow

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

\[\text{while } (x > 0)\]

\[x = x + y;\]
Programs as Graphs

```java
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;
}
```

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

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

Graph Neural Networks

- node selection
- node classification
- graph classification

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

https://github.com/microsoft/tf-gnn-samples/
Initial Node Representations

Label: outFilePrefix
Type: string

out, file, prefix
Split to subtokens

string, object, ...
All implemented types

Embed, , , ...
Average

Concat
Max Pool

Embed, , , ...

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

\[
f(h_T^{SLOT}, h_T^{first}) \gg f(h_T^{SLOT}, h_T^{clazz})
\]
Quantitative Results – Variable Misuse

<table>
<thead>
<tr>
<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>
</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, σ = 2.6)
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>
<tr>
<td>Unseen Projects</td>
<td>28.9</td>
<td>60.2</td>
<td>78.2</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.

JK +1
var x = ComputeX();
if (some-condition-without-x) {
  UseX(x)
}
else {
  UseOtherVars(y)
}
// x not used after this point
Types & Machine Learning
let $a = 1$; let $b = a + 1$;

DeepTyper  
Hellendoorn, Barr, Bird, Allamanis, 2018

Embedding Layer

bi-GRU

Variable Sync

Classification of Type

- type for $a$
- type for $b$
- type for $a$
Combining DeepTyper with CheckJS

CheckJS (classic typechecker) → Type Annotations (but contain any’s that can be refined)

Accuracy: 10.5%

DeepTyper predict on all >90% confidence → "Augmented" Type Annotations

Accuracy: 37.6%
False Positives: 1.2%

Hellendoorn, Barr, Bird, Allamanis, FSE 2018
Conceptual Types

"a password"  ->  string password;

"a JSON string"  ->  string data = Json.Load();

Latent; we don’t observe in 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

\[
\text{filePath} \\
\downarrow \\
\text{fileDir} \\
\downarrow \\
``../app.exe`` \\
\downarrow \\
\text{execDir}
\]

\[
\text{fileDir} = \text{filePath} \\
\text{execDir} = 
``../app.exe``
\]
Constructing Name Flows – Returns

```python
def NormalizePath(...):
    ...
    return normalized;
```
Constructing Name Flows – Function Calls

\[
\text{NormalizePath} \\
\downarrow \\
\text{normedPath} = \text{NormalizePath}(\ldots)
\]
Constructing Name Flows – Actuals to Formals

```python
pathToFile
  ↓
  path

```
def Exists(string path) { ... }

Exists(pathToFile)
Constructing Name Flows – Override

```
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

through information theory...

Dataflow of string variables/method and their names.

Nominal string refinements.
We want both
\[ P(\text{cluster}|\text{name}) \]
\[ P(\text{name}|\text{cluster}) \]
to have low entropy
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

\[ VI(N, \{c_1\}) < VI(N, \{c_2, c_3\}) \]
From Dataflow to Nominal Type Refinements

\[ C^* = \arg\min_C VI(N, C) \text{ s.t. } \exists R = (C, \subseteq) \]
Full name of nodes or constant values

1. path, Path, originalRequestPath, modifiedRequestPath, owinRequestPath, "/
2. contentPath, basePath, IViewEngineHost::ExpandPath, AspNetRootPathProvider::GetRootPath, "/
3. DiagnosticsConfiguration::GetNormalizedPath, NancyContext::ToFullPath, ModulePath
4. DefaultCulture, defaultCulture, cookieCulture, cultureLetters, name
5. earlyExitReason, "Requires Any Claim", "Requires Claims", "Requires Authentication", "Accept"
6. IObjectSerializer::Serialize, DefaultObjectSerializer::Serialize, JsonObject::ToString
7. SimpleJson::SerializeObject, HttpRequestCollection::toString
8. method, Method, "PUT", "POST", "PATCH", "OPTIONS", "HEAD", "GET", "DELETE"
9. value, cookieValue, sourceString, "SomeValue", cookieValueEncrypted, attemptedValue, decryptedValue, defaultValue
10. password, "password", realPassword, plainText, Password
11. HttpUtility::UrlDecode, HttpUtility::UrlPathEncode, HttpUtility::UrlEncodeUnicode, redirectUrl
12. fallbackRedirectUrl, url, path
<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>
CodeSearchNet

- A corpus of 6 millions functions with metadata.
- A small human-annotated set of relevance annotations.
- A semantic code search challenge.

https://github.com/github/CodeSearchNet
Closing Thoughts
Machine Learning
Learn Patterns from Rich Structure

• Capture human elements within code
• Make probabilistic predictions

Formal Methods
Reasoning

• Capture formal constraints
• Exact reasoning
Variable Misuse Task

Possible type-correct options: class, first

Not easy to catch with static analysis tools.

From Dataflow to Nominal Type Refinements
through information theory.

Nominal string refinements.

From string variable/method names.

Informal type refinements.

Machine Learning
Learn Patterns from Rich Structure

- Capture human elements within code
- Make probabilistic predictions

Formal Methods
Reasoning

- Capture formal constraints
- Exact reasoning

CodeSearchNet

- A corpus of 4 millions functions with metadatas
- A small human-annotated set of relevance annotations.
- A semantic code search challenge.

https://github.com/miltos1/CodeSearchNet

@miltos1

miltos.allamanis.com
Neural Message Passing

Current Neighbor States

Prepare “Message”

Summarize Received Information

Next Node State

Current Node State
Graph Neural Networks: Message Passing
GNNs: Synchronous Message Passing (All-to-All)