Understanding and Generating Source Code
... with Deep Learning

Marc Brockschmidt - MSR Cambridge
@mmjb86
Understanding and Generating Source Code ... with Deep Learning

Marc Brockschmidt - MSR Cambridge
@mmjb86

& MSR collaborators
& MSR Interns
& VS IntelliCode Team
Personal History: Termination Proving
public class Flatten {
    public static IntList flatten(TreeList list) {
        TreeList cur = list;
        IntList result = null;
        while (cur != null) {
            Tree tree = cur.value;
            if (tree != null) {
                IntList oldIntList = result;
                result = new IntList();
                result.value = tree.value;
                result.next = oldIntList;
                TreeList oldCur = cur;
                cur = new TreeList();
                cur.value = tree.left;
                cur.next = oldCur;
                oldCur.value = tree.right;
            } else cur = cur.next;
        }
    }
}
final class List {
    List n;
    public void appE(int i) {
        if (n == null) {
            if (i <= 0) return;
            n = new List();
            i--;
        }
        n.appE(i);
    }
}
public class Loop {
    public static void main(String[] a) {
        int i = 0;
        int j = a.length;
        while (i < j) {
            i += a[i].length();
        }
    }
}
void iterate() {
    L3 x = this.n;
    while (x != this)
        x = x.n;
}
Personal History: Termination Proving

\[
\begin{align*}
\text{while } i > 0 & \text{ do} \\
& i = i - 1 \\
& x = x + i \\
\text{done} \\
\text{while } x > 0 & \text{ do} \\
& x = x - 1 \\
\text{done}
\end{align*}
\]
Personal History: Termination Proving

System.out.println("Hello World!")
procedure insertion_sort(lst: Node)
  requires 1seg(lst, null) * lst != null
{
  var prv := null;
  var srt := lst;
  while (srt != null) {
    var curr := srt.next;
    var min := srt;
    while (curr != null) {
      if (curr.data < min.data)
        min := curr;
      curr := curr.next;
    }
    var tmp := min.data;
    min.data := srt.data;
    srt.data := tmp;
    prv := srt;
    srt := srt.next;
  }
}
procedure insertion_sort(lst: Node)
  requires lseg(lst, null) * lst != null
  { 
    var prv := null;
    var srt := lst;
    while (srt != null) { 
      var curr := srt.next;
      var min := srt;
      while (curr != null) { 
        if (curr.data < min.data) 
          min := curr;
        curr := curr.next;
      } 
      var tmp := min.data;
      min.data := srt.data;
      srt.data := tmp;
      prv := srt;
      srt := srt.next;
    }
  }

procedure insertion_sort(lst: Node)
  requires lseg(lst, null) * lst != null
  ensures lseg(lst, null) * lst != null
  { 
    var prv := null;
    var srt := lst;
    while (srt != null) { 
      var curr := srt.next;
      var min := srt;
      while (curr != null) { 
        invariant (prv == null * srt == lst
                   * lseg(lst, null))
        || (lseg(lst, prv) * prv.next = srt
            * lseg(srt, null)) 
        if (curr.data < min.data) 
          min := curr;
        curr := curr.next;
      } 
    }
  }
Team Overview

Program Structure
Team Overview

- Interpretable
- Generalisation verifiable

- Manual effort
- Limited to specialists
Team Overview

- 🔄 Understands images/language/speech
- Green: Finds patterns in noisy data

- Requires many samples
- Handling structured data is hard

- Interpretable
- Green: Generalisation verifiable

- Manual effort
- Limited to specialists
Team Overview

Deep Learning

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

Program Structure

- Interpretable
- Generalisation verifiable
  - Manual effort
  - Limited to specialists

Procedural Artificial Intelligence
Team Overview

Understanding Programs

Deep Learning

Procedural Artificial Intelligence

Program Structure
Team Overview

Deep Learning

Procedural Artificial Intelligence

Program Structure

Understanding Programs

Program-structured ML models
Team Overview

Deep Learning

Program Structure

Procedural Artificial Intelligence

Understanding Programs

Program-structured ML models

Generating Programs
Team Overview

Deep Learning

Procedural Artificial Intelligence

Program Structure

Generating Programs

Understanding Programs

Program-structured ML models
The Big Picture
Big Code: Potential
Big Code: Potential

Code Patterns
Big Code: Potential

- Code Patterns
- Natural Language
Big Code: Potential

- Code Patterns
- Natural Language
- Development Histories
```csharp
private static string NormalizePath(string path)
{
    path = path.Replace('\\', '/');

    if (path.[...])
    {
        return path;
    }

    bool string.EndsWith(string value)
    (+3 overloads)
    Determines whether the end of this string inst...
    ★ IntelliCode suggestion based on this context
```
private
{
    padding = new Rect(0, 0, 10, 10);
    // Set up the initial perimeter

    double x1 = b.Left - padding.Left;
    double x2 = b.Right + padding.Right;
    if (x1 < x2)
    {
        double y1 = b.TextTop - padding.Top;
        double y2 = b.TextBottom + padding.Bottom;
        newBounds.Add(new Rect(x1, y1, x2 - x1, y2 - y1));
    }
}

public static
{
    if (rectangle == null)
        return null;

    IntelliCode

    Did you mean to use y1 instead of x1? Suggested based on analysis of code patterns in this repo.

    Apply Fix

    Active
loss = tf.reduce_sum(tf.square(linear_model - y))  # sum of the squares
# optimizer
optimizer = tf.train.GradientDescentOptimizer(0.01)

train = optimizer.minimize(loss)
# training loop
init = tf.
# Core editorconfig formatting - indentation
6
7
8 # use soft tabs (spaces) for indentation
9 indent_style = space
10
11 # Formatting - indentation options
12
13 # indent switch case contents.
14 csharp_indent_case_contents = true
15 # indent switch labels
16 csharp_indent_switch_labels = true
17
18 # Formatting - new line options
19
20 # place catch statements on a new line
21 csharp_new_line_before_catch = true
22 # place else statements on a new line
23 csharp_new_line_before_else = true
24 # require finally statements to be on a new line after the closing brace
25 csharp_new_line_before_finally = true
26 # require braces to be on a new line for methods, types, lambdas, accessors, properties, object_collection, a
27 csharp_new_line_before_open_brace = methods, types, lambdas, accessors, properties, object_collection, contr
Understanding Programs
Task: Detecting Variable Misuse

Given location in program code, identify which variable should be used:

```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);
```
Learning from Programs

Learning from Graphs

Generating Programs

Task: Detecting Variable Misuse

Given location in program code, identify which variable should be used:

```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
Task: Detecting Variable Misuse

Given location in program code, identify which variable should be used:

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

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

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.
Task: Suggesting Good Variable Names

```cpp
int SumEven(int[] arr, int lim) {
    int sum = 0;
    for (int i = 0; i < lim; i++)
        if (arr[i] % 2 == 0)
            sum += arr[i];
    return sum;
}
```
Analysing Code: PL View
Analysing Code: PL View

Approach 1: Proving Software Correct

• Needs Specifications
• Limited Domains
• Limited Size

Approach 2: Finding Software Bugs

• Manual Error Pattern Definitions
• Hard to Configure
Analysing Code: ML View
Analysing Code: ML View

Approach 1.1: Sequence or tree of words
Analysing Code: ML View

Approach 1.1: Sequence or tree of words (re-using NLP ideas)

Programs are different from natural language:

• Semantics for keywords already known
• Many words (APIs, local methods) only used seldomly
• Long-distance dependencies common

Approach 2: Graphs

• Nodes labelled by semantic information
• Edges for semantic relationships
Analysing Code: ML View

Approach 1.1: Sequence or tree of words (re-using NLP ideas)

Programs are different from natural language:
- Semantics for keywords already known
- Many words (APIs, local methods) only used seldomly
- Long-distance dependencies common

Approach 2: Graphs
- Nodes labelled by semantic information
- Edges for semantic relationships

From Static Analysis
Programs as Graphs: Syntax

Assert.NotNull(clazz);
Programs as Graphs: Syntax

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

Next Token

```
Assert
.  
NotNull
( 
... 
```
Programs as Graphs: Syntax

```
Assert.NotNull(clazz);
```
(x, y) = Foo();

while (x > 0)
    x = x + y;
Programs as Graphs: Data Flow

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

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

\[x = x + y;\]
Programs as Graphs: Data Flow

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

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

\[
x = x + y;
\]
Programs as Graphs: Data Flow

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

while \( x > 0 \)

\[
x = x + y;
\]
Programs as Graphs: Node Representation

Label: outFilePrefix
Type: string
Programs as Graphs: Node Representation

Label: outFilePrefix
Type: string
Programs as Graphs: Node Representation

Label: outFilePrefix
Type: string
Programs as Graphs: Node Representation

Label: outFilePrefix
Type: string

Split to subtokens

out, file, prefix

Embed

, , ,

Average

All implemented types

string, object, ...

Generating Programs
Programs as Graphs: Node Representation

Label: outFilePrefix
Type: string

out, file, prefix

Embed

Split to subtokens

Average

All implemented types

Max Pool

string, object, ...

Embed

[,],[,]
Programs as Graphs: Node Representation

Label: out
Type: string

split to subtokens

out, file, prefix

Embed

Average

Concat

All implemented types

string, object, ...

Embed

Max Pool
Programs as Graphs

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

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

\[x = x + y;\]

In practice: ~3000 nodes/graph, ~10000 edges/graph
Graph Neural Networks: Extending RNNs
Graph Neural Networks: Extending RNNs

Chain structured data
(e.g. text)
Graph Neural Networks: Extending RNNs

Chain structured data
(e.g. text)

Recurrent unit
Graph Neural Networks: Extending RNNs

embed('the') =  

Chain structured data  
(e.g. text)  

Recurrent unit
Graph Neural Networks: Extending RNNs

\[ \text{embed('the')} = \begin{array}{c} \text{the cat in the hat} \end{array} \]

Chain structured data (e.g. text)

\[ \text{Recurrent unit} \]
Graph Neural Networks: Extending RNNs

$\text{embed('the')} = \begin{array}{c} \square \end{array}$

Chain structured data
(e.g. text)

Recurrent unit
Graph Neural Networks: Extending RNNs

embed('the') = □

Chain structured data (e.g. text)

Recurrence unit
Graph Neural Networks: Extending RNNs

$\text{embed('the')} = \square$

the → cat → in → the → hat

Chain structured data (e.g. text)

△ Recurrent unit
Graph Neural Networks: Extending RNNs

```
embed('the') = [0, 0, 1, 0, 0, 0, 0, 0, 0, 0, 0, 0]

the => cat => in => the => hat

Chain structured data (e.g. text)

• Recurrent unit

expr' = △ (expr, expr)
```
Graph Neural Networks: Extending RNNs

\[ \text{embed('the')} = \text{the cat in the hat} \]

Chain structured data
(e.g. text)

\[ \text{Recurrence unit} \]

\[ \text{The symbol} = \text{representation of the symbols} (, ) \]
Graph Neural Networks: Extending RNNs

embed('the') = [ ]

the → cat → in → the → hat

Chain structured data (e.g. text)

Recurrent unit

let recurrent_unit state input = ... in foldl recurrent_unit init_state seq

<textarea>
embed('the') = [ ]

the → cat → in → the → hat

Chain structured data (e.g. text)

Recurrent unit

let recurrent_unit state input = ... in foldl recurrent_unit init_state seq
</textarea>
Graph Neural Networks: Extending RNNs

embed('the') = 🗔

Chain structured data (e.g. text)

Recurrent unit

let recurrent_unit state input = ... in
foldl recurrent_unit init_state seq

let △ ▼ ▼ ▼ ▼ = ... in
foldl △ ▼ ▼ ▼ ▼ [ ▼ , ..., ▼ ]
Graph Neural Networks: States
Graph Neural Networks: States

label embedding

\begin{center}
\begin{tabular}{c|c|c|c|c|c}
  \hline
  \text{Label Embedding} & -0.7 & 12 & 0.18 & -9 & \ldots & 0.32 \\
  \hline
\end{tabular}
\end{center}
Graph Neural Networks: States

Label Embedding

\[
\begin{array}{cccc}
-0.7 & 12 & 0.18 & -9 \\
\ldots & \ldots & \ldots & \ldots
\end{array}
\]

\[\Rightarrow \text{Envelope}\]
Graph Neural Networks: States

Label Embedding

<table>
<thead>
<tr>
<th>Label</th>
<th>Value</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>-.7</td>
</tr>
<tr>
<td>2</td>
<td>12</td>
</tr>
<tr>
<td>3</td>
<td>.18</td>
</tr>
<tr>
<td>4</td>
<td>-9</td>
</tr>
<tr>
<td>5</td>
<td>...</td>
</tr>
<tr>
<td>6</td>
<td>.32</td>
</tr>
</tbody>
</table>

$\text{Label Embedding} = \begin{bmatrix} -0.7 & 12 & 0.18 & -9 & \ldots & 0.32 \end{bmatrix}$
Graph Neural Networks: States

Edge Type 1 / $NN_1$
Graph Neural Networks: States

Edge Type 1 / $NN_1$

Edge Type 2 / $NN_2$
Graph Neural Networks: Propagation

- $NN_1$
- $NN_2$
- Recurrent unit
Graph Neural Networks: Propagation

- $NN_1$
- $NN_2$
- Recurrent unit
Graph Neural Networks: Propagation

- $NN_1$
- $NN_2$
- Recurrent unit
Graph Neural Networks: Propagation

- $NN_1$
- $NN_2$
- Recurrent unit
Graph Neural Networks: Propagation

$ NN_1$

$ NN_2$

Recurrent unit

$ \text{node} \, ' = \bigtriangleup ( \text{node}, \sum \text{neighbor} )$
Graph Neural Networks: Unrolling
Graph Neural Networks: Unrolling
Graph Neural Networks: Uses
Graph Neural Networks: Uses

Gated Graph Sequence Neural Networks. In ICLR’16.
Graph Neural Networks: Implementation
Graph Neural Networks: Implementation

Train Performance:
On Titan X: 250 000 nodes/s (80 graphs/s)
On V100: 750 000 nodes/s (250 graphs/s)

Test Performance:
On Titan X: 660 000 nodes/s (220 graphs/s)
On V100: 1 350 000 nodes/s (450 graphs/s)
Detecting Variable Misuse
Detecting Variable Misuse

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

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

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

**Objective:** Given representation of SLOT, choose between “first” and “clazz”
## Variable Naming: Quantitative Results

<table>
<thead>
<tr>
<th>F1 (%)</th>
<th>Sequence</th>
<th>Seq.+Dataflow</th>
<th>Graph</th>
</tr>
</thead>
<tbody>
<tr>
<td>Seen Projects</td>
<td>44.0</td>
<td>50.1</td>
<td>65.8</td>
</tr>
<tr>
<td>Unseen Projects</td>
<td>30.6</td>
<td>32.0</td>
<td>62.0</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
## Variable Misuse: Quantitative Results

<table>
<thead>
<tr>
<th>Accuracy (%)</th>
<th>Sequence</th>
<th>Seq.+Dataflow</th>
<th>Graph</th>
</tr>
</thead>
<tbody>
<tr>
<td>Seen Projects</td>
<td>50.0</td>
<td>73.7</td>
<td>86.5</td>
</tr>
<tr>
<td>Unseen Projects</td>
<td>28.9</td>
<td>60.2</td>
<td>82.0</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)
# Variable Misuse: Quantitative Results

<table>
<thead>
<tr>
<th>Accuracy (%)</th>
<th>Sequence</th>
<th>Seq.+Dataflow</th>
<th>Graph</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Seen Projects</strong></td>
<td>50.0</td>
<td>73.7</td>
<td>86.5</td>
</tr>
<tr>
<td><strong>Unseen Projects</strong></td>
<td>28.9</td>
<td>60.2</td>
<td>82.0</td>
</tr>
<tr>
<td>255 Proj. – Seen</td>
<td>-</td>
<td>-</td>
<td>91.8</td>
</tr>
<tr>
<td>255 Proj. – Unseen</td>
<td>-</td>
<td>-</td>
<td>89.4</td>
</tr>
</tbody>
</table>
Task: Extracting Best Practices

**Objective:** Given many commits, extract common kinds of changes
Task: Extracting Best Practices

Objective: Given many commits, extract common kinds of changes

```csharp
emps.Where(e => m.ReportsTo == e.EmployeeID).FirstOrDefault();
emps.FirstOrDefault(e => m.ReportsTo == e.EmployeeID);

users.Where(u => u.Item1 == username && u.Item2 == password).FirstOrDefault();
users.FirstOrDefault(u => u.Item1 == username && u.Item2 == password);

typ = source == null ? typeof(object) : source.GetType();
typ = source?.GetType() ?? typeof(object);
```

```csharp
sources = sources == null ? new object[0] : sources.ToArray();
sources = sources?.ToArray() ?? new object[0];
```
**Task: Extracting Best Practices**

**Objective:** Given many commits, extract common kinds of changes

**Idea:** Learn to embed similar diffs nearby in vector space (as in word2vec)

```csharp
emps.Where(e => m.ReportsTo == e.EmployeeID).FirstOrDefault();
emps.FirstOrDefault(e => m.ReportsTo == e.EmployeeID);
users.Where(u => u.Item1 == username && u.Item2 == password).FirstOrDefault();
users.FirstOrDefault(u => u.Item1 == username && u.Item2 == password);
```

```csharp
sources = sources == null ? new object[0] : sources.ToArray();
sources = sources?.ToArray() ?? new object[0];
```

```csharp
typ = source == null ? typeof(object) : source.GetType();
typ = source?.GetType() ?? typeof(object);
```

```csharp
sources = sources == null ? new object[0] : sources.ToArray();
sources = sources?.ToArray() ?? new object[0];
```
Task: Extracting Best Practices

**Objective:** Given many commits, extract common kinds of changes

**Idea:** Learn to embed similar diffs nearby in vector space (as in word2vec)
Learning From Programs: Key Points

**Insight:** GNNs successful at learning with code semantics

**Outcomes:**
- Machinery can be re-used for many tasks
- Learns “soft” rules from data, no rule definitions required
- Found number of bugs in mature code
Generating Programs
Task: Filling in Blanks

Given location in program code, generate expression:

```csharp
int methParamCount = 0;
if (paramCount > 0) {
    IParameterTypeInfo[] moduleParamArr =
        GetParamTypeInfo(Dummy.Signature, paramCount);
    methParamCount = moduleParamArr.Length;
}
if (/* blank */) {
    IParameterTypeInfo[] moduleParamArr =
        GetParamTypeInfo(Dummy.Signature, paramCount - methParamCount);
}
```
Task: Filling in Blanks

Given location in program code, generate expression:

```csharp
int methParamCount = 0;
if (paramCount > 0) {
    IParameterTypeInfo[] moduleParamArr = 
        GetParamTypeInfo(Dummy.Signature, paramCount);
    methParamCount = moduleParamArr.Length;
}
if (paramCount > methParamCount) {
    IParameterTypeInfo[] moduleParamArr = 
        GetParamTypeInfo(Dummy.Signature, 
                         paramCount - methParamCount);
}
```
Overview of Approach

Program with hole → GNN → Context Representation
Overview of Approach

Program with hole $\xrightarrow{\text{GNN}}$ Context Representation $\xrightarrow{\text{Initialization}}$ AST Node

Learning from Programs
Learning from Graphs
Generating Programs
Overview of Approach

Program with hole → GNN → Context Representation → Initialization → AST Node → Node 1 → ... → Node K → Tree Generation
Overview of Approach

Program with hole

GNN

Context Representation

Initialization

AST Node

Node 1

Node K

Variable

In-scope Variable 1

In-scope Variable 2

In-scope Variable 3

Choice
Generating Trees

Variables in scope
Generating Trees

Variables in scope

Expression

Expression

-
Generating Trees

Variables in scope

Expression
- Expression

Expression
Generating Trees

Variables in scope

\[ i \]

\[ j \]

Expression

- 

Expression
Generating Trees

Variables in scope

Expression - Expression

Expression

Expression

Expression

i

j
Generating Trees

Variables in scope

\[ i \]

\[ j \]

Expression

\[ i \]

Expression

\[ + \]

Expression

\[ - \]

Expression

\[ i \]
Generating Trees

Variables in scope

\[ i \]

\[ j \]

Expression

\[ i \]

Expression

\[ j \]

Expression
Generating Trees

Variables in scope

\( i \)

\( j \)
Generating Graphs

Variables in scope

\[ \text{i} \]
\[ \text{j} \]

AST Child

Expression

\[ \text{i} \]

Expression

\[ \text{j} \]
\[ 1 \]
Generating Graphs

Variables in scope

\[ i \]
\[ j \]

Expression

\[ - \]

Expression

\[ + \]

Expression

\[ 1 \]

AST Child

Next Token
Generating Graphs

Variables in scope

AST Child

Next Token

Next Use
Generating Graphs (with Attribute Grammars)

AST Child

Next Token

Next Use
Generating Graphs (with Attribute Grammars)

- Variables in scope
  - $i$
  - $j$

- Displayed graph with nodes and edges representing expressions and their relationships.
Generating Graphs (with Attribute Grammars)

Variables in scope

- \( i \)
- \( j \)

AST Child -> AST Parent

- Next Token
- Next Sibling
- Next Use
### Filling in Blanks: Quantitative Results

<table>
<thead>
<tr>
<th>Model</th>
<th>Perplexity</th>
<th>Type-Correct</th>
<th>Match@1</th>
<th>Match @5</th>
</tr>
</thead>
<tbody>
<tr>
<td>Seq → NAG</td>
<td>8.38</td>
<td>40.4</td>
<td>8.4</td>
<td>15.8</td>
</tr>
<tr>
<td>Graph → Tree</td>
<td>5.37</td>
<td>41.2</td>
<td>19.9</td>
<td>36.8</td>
</tr>
<tr>
<td>Graph → Syntax Networks</td>
<td>3.03</td>
<td>74.7</td>
<td>32.4</td>
<td>48.1</td>
</tr>
<tr>
<td>Graph → Sequentialised Tree</td>
<td>3.48</td>
<td>84.5</td>
<td>36.0</td>
<td>52.7</td>
</tr>
<tr>
<td>Graph → Neural Attr. Gram.</td>
<td>3.07</td>
<td>84.5</td>
<td>38.8</td>
<td>57.0</td>
</tr>
</tbody>
</table>

Training data: 479 C# projects from GitHub
Test data: 114 C# projects from GitHub (~100 000 samples)
UX Lessons Learned
Dogfooding Tales: The Good

// 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}");

smartbot@microsoft.com  1/31/2018  Update 1
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.

John Keech  1/31/2018
+1
Dogfooding Tales: The Good
Dogfooding Tales: The Strange

```csharp
string activeRepo = this.gitExt.ActiveRepositories[0].RepositoryPath;
string relativePath = PathHelper.MakeRelative(activeRepo, sourceFileName)
Directory.CreateDirectory(Path.GetDirectoryName(compositePath));

try
```

smartbot@microsoft.com 31 minutes ago
Based on this repo's code patterns, did you intend to use 'compositePath' (confidence 72%) rather than 'sourceFileName' (confidence 11%) here? Review is recommended by Research bot's Variable Misuse analysis.

Kenny Young 25 minutes ago
relativePath is correct here, though I understand why this code path is a bit tricky for the bot - here we are building the path to pass to the Git API to read the older version of the file. compositePath is the output path, appended with the hash.

Kenny Young 18 minutes ago
Oops, I meant "sourceFileName is correct here". Same argument. Does the Variable Misuse analyzer search PR comments? 😊

Kenny Young 10 minutes ago
I'm actually going to take this comment to mean "hey, this code is hard to read" and move the CreateDirectory line above this code, so that like variables are used together. That will surely unconfuse the bot and be easier to read as well.
Dogfooding Tales: The Bad

```csharp
UnhandledExceptionReporterTests.cs

[Fact]
public async Task ExceptionHandler.Validate_Production_Returns_Empty()
{
    using (var telemetryWriter = new StringWriter(new StringBuilder(), CultureInfo.InvariantCulture))
    {
        var logger = DiagnosticsLogger.New(new LogValueSet(), telemetryWriter);
        var errorMessage = "ThisIsATest";
    }
}
```

smartbot@microsoft.com 1/31/2018
Research Bot suggests renaming `telemetryWriter` as `w` with confidence 79%.
Understanding and Generating Source Code

**Question:** How to learn from code with semantics?

**Hypothesis:** Code is natural, targets people and machines

**Our Solution:** Graphs representing all modalities
Understanding and Generating Source Code

**Question:** How to learn from code with semantics?

**Hypothesis:** Code is natural, targets people and machines

**Our Solution:** Graphs representing all modalities

Marc Brockschmidt
@mmjb86