Understanding Code using Natural Language & Graph Neural Networks

Miltos Allamanis

Microsoft Research, Cambridge

@miltos1

miltos.allamanis.com

Joint work with Marc Brockschmidt, Patrick Fernandes, Mahmoud Khademi, Hamel Husain, Ho-Hsiang Wu, Tiferet Gazit
Deep Program Understanding

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

- Interpretable
- Generalisation verifiable
  - Manual effort
  - Limited to specialists
Source Code and Natural Language
Code Autocompletion

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”
Inferring Type Refinements

Conceptual Types

"a password"  →  string password;

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

Defined Types

Latent; we don’t observe in the conceptual types.

Defined explicitly by the programmer.

Dash et al. 2018 “RefiNym: Using Names to Refine Types”
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
Research in ML+Code

- Infer latent intent
- Ambiguous information

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.

CCS Concepts: - Computing methodologies → Machine learning; Natural language processing; - Software → Static analysis; Program analysis;
- Graph Neural Networks
- Understanding Code
- Structured Summarization
- Code Search
Graph Neural Networks

and Neural Message Passing
Graph Neural Networks

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

Neural Message Passing

Current Neighbor States

Prepare “Message”

Summarize Received Information

Next Node State

Current Node State
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/tf-gnn-samples/
Understanding & Generating Source Code

...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;
}
```
Assert.NotNull(clazz);
Programs as Graphs: Data Flow

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

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)
            sum += arr[i];
    }
    return sum;
}

~900 nodes/graph  ~8k edges/graph
Initial Node Representations

Label: outFilePrefix
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.
```
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

\[
f(h_T^{SLOT}, h_T^{first}) \gg f(h_T^{SLOT}, h_T^{clazz})
\]
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>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>
<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.
var x = ComputeX();
if (some-condition-without-x) {
    UseX(x);
} else {
    UseOtherVars(y);
}
// x not used after this point
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;
}

Summary

Returns the sum of the positive numbers in an array

Fernandes et al. 2019 “Neural Structured Summarization”
**SeqGraph2Seq**

\[
\tilde{h}^{(0)}_n = \begin{cases} 
ENCODER(P)_n & \text{if } n \in P \\
h^{(0)}_n & \text{else}
\end{cases}
\]

\[
h^{(T)}_n = biGNN \left( G, \left\{ \tilde{h}^{(0)}_1, ..., \tilde{h}^{(0)}_N \right\} \right)_n
\]

*P is the backbone sequence*
Graph to Sequence Model

Encoder

Decoder

Pool

$[h_A^T, ..., h_G^T]$
## Quantitative Results

<table>
<thead>
<tr>
<th>C# Documentation</th>
<th>F1</th>
<th>ROUGE-2</th>
<th>ROUGE-L</th>
</tr>
</thead>
<tbody>
<tr>
<td>biRNN -&gt; RNN</td>
<td>35.2</td>
<td>20.8</td>
<td>36.7</td>
</tr>
<tr>
<td>GNN -&gt; RNN</td>
<td>38.9</td>
<td>25.6</td>
<td>37.1</td>
</tr>
<tr>
<td>biRNN + GNN -&gt; RNN</td>
<td>45.4</td>
<td>28.3</td>
<td>41.1</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Java Method Naming</th>
<th>F1</th>
<th>ROUGE-2</th>
<th>ROUGE-L</th>
</tr>
</thead>
<tbody>
<tr>
<td>Alon et al. *</td>
<td>43.0</td>
<td>-</td>
<td>-</td>
</tr>
<tr>
<td>biRNN -&gt; RNN</td>
<td>35.8</td>
<td>17.9</td>
<td>39.7</td>
</tr>
<tr>
<td>biRNN + GNN -&gt; RNN</td>
<td>44.7</td>
<td>21.1</td>
<td>43.1</td>
</tr>
</tbody>
</table>

Gunshots were fired at rapper Lil Wayne’s tour bus early Sunday in Atlanta. No one was injured in the shooting, and no arrests have been made, Atlanta Police spokeswoman Elizabeth Espy said. Police are still looking for suspects. Officers were called to a parking lot in Atlanta’s Buckhead neighbourhood, Espy said. They arrived at 3:25 a.m. and located two tour buses that had been shot multiple times. The drivers of the buses said the incident occurred on Interstate 285 near Interstate 75, Espy said. Witnesses provided a limited description of the two vehicles suspected to be involved: a “Corvette-style vehicle” and an SUV. Lil Wayne was in Atlanta for a performance at Compound nightclub Saturday night. CNN’s Carma Hassan contributed to this report.

Rapper Lil Wayne not injured after shots fired at his tour bus on an Atlanta interstate, police say. No one has been arrested in the shooting.
Structure in Natural Language
Quantitative Results

<table>
<thead>
<tr>
<th>CNN/DailyMail</th>
<th>ROUGE-1</th>
<th>ROUGE-2</th>
<th>ROUGE-L</th>
</tr>
</thead>
<tbody>
<tr>
<td>RNN -&gt; RNN *</td>
<td>31.3</td>
<td>11.8</td>
<td>28.8</td>
</tr>
<tr>
<td>RNN + GNN -&gt; RNN</td>
<td>33.0</td>
<td>13.3</td>
<td>28.3</td>
</tr>
</tbody>
</table>

* Encoder -> Decoder

* See et al. (2015). *Get To The Point: Summarization with Pointer-Generator Networks*
Quantitative Results

<table>
<thead>
<tr>
<th>CNN/DailyMail</th>
<th>ROUGE-1</th>
<th>ROUGE-2</th>
<th>ROUGE-L</th>
</tr>
</thead>
<tbody>
<tr>
<td>RNN -&gt; RNN *</td>
<td>31.3</td>
<td>11.8</td>
<td>28.8</td>
</tr>
<tr>
<td>RNN + GNN -&gt; RNN</td>
<td>33.0</td>
<td>13.3</td>
<td>28.3</td>
</tr>
<tr>
<td>RNN -&gt; RNN + pointer *</td>
<td>36.4</td>
<td>15.7</td>
<td>33.4</td>
</tr>
<tr>
<td>RNN + GNN -&gt; RNN + pointer</td>
<td>38.1</td>
<td>16.1</td>
<td>33.2</td>
</tr>
</tbody>
</table>

* See et al. (2017). Get To The Point: Summarization with Pointer-Generator Networks
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
NLP with GNNs

Grounded Language

Procedural Text

Semantic Parsing
Source Code & Natural Language

Rich Structure

Reasoning
Neural Message Passing

Programs as Graphs: Data Flow

Graph to Sequence Model

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