Augmented Programmer Intelligence

Eran Yahav
Technion
# Analysis and Synthesis with “Big Code”

Programming Languages + ML + IR + …

<table>
<thead>
<tr>
<th>Project</th>
<th>Conference</th>
</tr>
</thead>
<tbody>
<tr>
<td>Code2vec: Learning Distributed Representations of Code</td>
<td>POPL’19</td>
</tr>
<tr>
<td>On the Practical Computational Power of Finite Precision RNNs for Language Recognition</td>
<td>ACL’18</td>
</tr>
<tr>
<td>Extracting Automata from Recurrent Neural Networks Using Queries and Counterexamples</td>
<td>ICML’18</td>
</tr>
<tr>
<td>A General Path-based Representation for Predicting Program Properties</td>
<td>PLDI’18</td>
</tr>
<tr>
<td>Synthesis of Forgiving Data Extractors</td>
<td>WSDM’17</td>
</tr>
<tr>
<td>Lossless Separation of Web Pages into Layout Code and Data</td>
<td>KDD’16</td>
</tr>
<tr>
<td>Statistical Similarity of Binaries</td>
<td>PLDI’16</td>
</tr>
<tr>
<td>Estimating Types in Stripped Binaries</td>
<td>POPL’16</td>
</tr>
<tr>
<td>…</td>
<td></td>
</tr>
<tr>
<td>Synthesis from partial programs</td>
<td>OOPSLA’12</td>
</tr>
</tbody>
</table>

[GitHub](https://github.com/tech-srl) | [Code2vec.org](http://code2vec.org)

PRIME
TRACY
DIZY
Like2Drops

[Codota](https://www.codota.com)

Gail Weiss, Uri Alon, Hila Peleg, Meital Zilberstein, Omer Katz, Yaniv David, Adi Omari, Shir Yadid, Nimrod Partush, Dana Drachsler
 Lots of code available on the web

Learn from all the code out there to make software development faster and smarter

27M users
89M repositories

9M registered users
16M questions
25M answers
Why now?

Big code

Static program analysis

Machine learning

Computation resources
Augmented Programmer Intelligence

- Predict code (preventive software quality)
  - Automate mundane tasks
  - Keep devs on the main path
  - “likely by construction”

- Check code
  - Standardize on practices learned from code
From programs to models and back

- Neural networks for predicting program elements

- What is it that a network has learned?
  - A model that provides some explanation
  - A new technique for extracting explanation (DFA) from a recurrent neural network

- What is it that a network can learn?
**Code2vec [POPL’19]**

- A neural network for **predicting program elements from context**
- Learns name embeddings, path embeddings, and simultaneously learns how to aggregate them
- **Example:** predicting method names = ~14M training methods,
Motivating example #1 (code2vec)

```java
String[] f(final String[] array) {
    final String[] newArray = new String[array.length];
    for (int index = 0; index < array.length; index++) {
        newArray[array.length - index - 1] = array[index];
    }
    return newArray;
}
```

<table>
<thead>
<tr>
<th>Method</th>
<th>Score</th>
</tr>
</thead>
<tbody>
<tr>
<td>reverseArray</td>
<td>77.34%</td>
</tr>
<tr>
<td>reverse</td>
<td>18.18%</td>
</tr>
<tr>
<td>subArray</td>
<td>1.45%</td>
</tr>
<tr>
<td>copyArray</td>
<td>0.74%</td>
</tr>
</tbody>
</table>
Motivating example #2 (code2seq)

```csharp
void Main() {
    string text = File.ReadAllText(@"T:\File1.txt");
    int num = 0;
    text = (Regex.Replace(text, "map", delegate (Match m) {
               return "map" + num++;
            }));
    File.WriteAllText(@"T:\File1.txt", text);
}
```

replace a string in a text file
Variety of applications

- Captioning
- Predict descriptive names
- Code review for existing names
- Automatic documentation
- Code retrieval
Background: Distributed Representations ("embeddings")

"One-hot" vectors

<table>
<thead>
<tr>
<th>Word</th>
<th>Vector</th>
</tr>
</thead>
<tbody>
<tr>
<td>hello</td>
<td>[0,0,...,0,1,0,0]</td>
</tr>
<tr>
<td>world</td>
<td>[0,0,...,0,1,0,0]</td>
</tr>
<tr>
<td>how</td>
<td>[0,0,...,0,1,0,0]</td>
</tr>
<tr>
<td>are</td>
<td>[0,0,...,0,1,0,0]</td>
</tr>
<tr>
<td>you</td>
<td>[0,0,...,0,1,0,0]</td>
</tr>
<tr>
<td>hi</td>
<td>[0,0,...,0,1,0,0]</td>
</tr>
</tbody>
</table>

Distributed representations

<table>
<thead>
<tr>
<th>Word</th>
<th>Vector</th>
</tr>
</thead>
<tbody>
<tr>
<td>hello</td>
<td>[0,0,...,0,1,0,0]</td>
</tr>
<tr>
<td>world</td>
<td>[0,0,...,0,1,0,0]</td>
</tr>
<tr>
<td>how</td>
<td>[0,0,...,0,1,0,0]</td>
</tr>
<tr>
<td>are</td>
<td>[0,0,...,0,1,0,0]</td>
</tr>
<tr>
<td>you</td>
<td>[0,0,...,0,1,0,0]</td>
</tr>
<tr>
<td>hi</td>
<td>[0,0,...,0,1,0,0]</td>
</tr>
</tbody>
</table>

Each vector: (0,0,...,0,1,0,0)

sim("hello","hi") > sim("hello","you")
Neural Networks

- A sequence of simple algebraic functions over vectors and matrices
- **Example**: Predict a how positive is a sentence (regression)

```plaintext
vec ← vec − α \frac{\partial \text{loss}}{\partial vec}

w ← w − α \frac{\partial \text{loss}}{\partial w}
```
AST paths: a general method to represent code in machine learning models

while (!done) {
    if (someCondition()) {
        done = true;
    }
}

(done,)

Code snippet represented as a set of all its syntactic paths
Network architecture

(source1, path, target1)

(source2, path, target2)

(source_n, path, target_n)

Bag of contexts
Works great, but

• Monolithic labels
• Monolithic paths
• Huge vocabulary

• *How can we do better?*
Long Short-Term Memory (LSTM)

- A kind of Recurrent Neural Networks
  - Input: vector
  - Updates its internal memory vector
  - Output: vector
- Input: a sequence of vectors
- Output: a sequence of vectors
- Extremely good at learning sequences
- The basis for (almost) any model that learns sequences (e.g., machine translation, speech recognition, image captioning...)

http://colah.github.io/posts/2015-08-Understanding-LSTMs/
Sequence-to-sequence (seq2seq) with LSTMs

• A basic approach:
  • LSTM encoder
Sequence-to-sequence (seq2seq) with LSTMs

• A basic approach:
  • LSTM encoder
  • LSTM decoder

how are you

cómo estás
Attention

- The decoder can “focus” on different inputs at each step.
  - By computing a dynamic weighted-average of the encoder states

[“Neural Machine Translation by Jointly Learning to Align and Translate”, Bahdanau et al., ICLR’2015]
[“Effective Approaches to Attention-based Neural Machine Translation”, Luong et al., EMNLP’2015]
Attention

- The decoder can “focus” on different inputs at each step.
  - By computing a dynamic weighted-average of the encoder states

[“Neural Machine Translation by Jointly Learning to Align and Translate”, Bahdanau et al., ICLR’2015]
[“Effective Approaches to Attention-based Neural Machine Translation”, Luong et al., EMNLP’2015]
Back to our problem...
The direct approach - treat code as text

Results are generally poor

void Main {
  string text = File ...

["Summarizing Source Code using a Neural Attention Model", Iyer et al., ACL'2016]
["A Neural Architecture for Generating Natural Language Descriptions ...", Loyola et al., ACL'2017]
```csharp
void Main()
{
    string text = File.ReadAllText(@"T:\File1.txt");
    int num = 0;
    text = (Regex.Replace(text, "map", delegate (Match m) {
        return "map" + num++;
    }));
    File.WriteAllText(@"T:\File1.txt", text);
}
```

replace a string in a text file

how to read a text file in c#

how to read a text file from a text file

code2seq (this work)

[CodeNN, ACL'2016]

seq2seq BiLSTMs
How to encode source code?

- Want an encoder that can capture the essence of a code snippet
  - Leverage the (known) syntax
  - Without losing language-generality
Key idea #1: use AST paths

```
while (!done) {
    if (someCondition()) {
        done = true;
    }
}
```

(done, SymbolRef)

[A General Path-based Representation for Predicting Program Properties, Alon et al., PLDI'2018]
A General Path-based Representation for Predicting Program Properties (PLDI’2018)

```python
def sh3(cmd):
    process = Popen(cmd, stdout=PIPE, stderr=PIPE, shell=True)
    out, err = process.communicate()
    retcode = process.returncode
    if retcode:
        raise CalledProcessError(retcode, cmd)
    else:
        return out.rstrip(), err.rstrip()
```

- Multiple tasks: predicting variable names, types
- Across programming languages: Java, JavaScript, Python, C#
- Multiple learning algorithms: e.g., use AST paths as factors in CRF

```java
import org.apache.hadoop.hbase.client.Connection;
```

```python
Configuration f = HBaseConfiguration.create();
Connection c =ConnectionFactory.createConnection(f);
```
Key idea #2: code snippet as aggregation of its paths

- [www.code2vec.org](http://www.code2vec.org)
- [https://github.com/tech-srl/code2vec](https://github.com/tech-srl/code2vec)
- Very fast to train (14M examples in ~24 hours).
- Large path, token and target vocabularies (of up to ~1M)
- But fits easily on an old K80 GPU

[code2vec: Learning Distributed Representations of Code, POPL ’2019]
code2vec

• Pros
  • General: can train on any pair of (code, target label)
    • Predict variable names, next token, yes/no malware etc.
  • Fast to train

• Cons
  • Data-hungry due to sparsity of paths
  • Cannot generate natural language sequences
Key idea #3: efficient encoding of paths

```
int countOccurrences(String str, char ch) {
    int num = 0;
    int index = -1;
    do {
        index = str.indexOf(ch, index + 1);
        if (index >= 0) {
            num++;
        }
    } while (index >= 0);
    return num;
}
```

```
int countOccurrences(String source, char value) {
    int count = 0;
    for (int i = 0; i < source.length(); i++) {
        if (source.charAt(i) == value) {
            count++;
        }
    }
    return count;
}
```
=> Encode paths in a more efficient way
Encode paths using an LSTM that "walks" on the AST

$k(=200)$ contexts are randomly sampled every iteration
code2seq - 50% more accurate, 10x smaller

Encode('getValue') = vec('get') + vector('value')

\[
\text{Encode('getValue')} = \text{vec('get')} + \text{vector('value')}
\]

\[
\text{concat}(\text{path, token, token})
\]

\[
\text{fully\textunderscore connected}(x) = \tanh(W \cdot x)
\]
Experiments: tasks

• Generating descriptions from StackOverflow (code, description) pairs (code captioning)
  • Dataset: C#, introduced by CodeNN, Iyer et al. (ACL'2016)
  • Especially challenging: only 53K examples
  • Target length: ~10 symbols avg.

• Generating method names (code summarization)
  • Datasets: 3 Java datasets (2 of them are new!) of sizes [0.7M, 4M, 16M] examples
  • Target length: ~3 symbols avg.
Baselines

• **Programming Language (PL) Baselines**
  • ConvAttention (Allamanis et al., ICML'2016)
  • Paths+CRFs (Alon et al., PLDI'2018)
  • Code2vec (Alon et al., POPL’2019)
  • CodeNN (Iyer et al., ACL'2016)

• **Neural Machine Translation (NMT) baselines**
  • Bidirectional, multi-layered LSTMs with attention
  • Transformer (Vaswani et al., NIPS'2017)
C# StackOverflow description prediction (code captioning)

**BLEU for the code captioning task (C#)**

<table>
<thead>
<tr>
<th>Model</th>
<th>BLEU</th>
</tr>
</thead>
<tbody>
<tr>
<td>MOSES (Koehn et al., 2007)</td>
<td>11.57</td>
</tr>
<tr>
<td>IR (Rush et al., 2015)</td>
<td>13.66</td>
</tr>
<tr>
<td>SUM-NN (Rush et al., 2015)</td>
<td>19.31</td>
</tr>
<tr>
<td>2-layer BiLSTM (split tokens)</td>
<td>19.78</td>
</tr>
<tr>
<td>Transformer (Vaswani et al., 2017)</td>
<td>19.68</td>
</tr>
<tr>
<td>CodeNN (Iyer et al., 2016)</td>
<td>20.53</td>
</tr>
<tr>
<td>code2seq</td>
<td><strong>23.04</strong></td>
</tr>
</tbody>
</table>
Java method name prediction (code summarization)

<table>
<thead>
<tr>
<th>Model</th>
<th>Java-small</th>
<th>Java-med</th>
<th>Java-large</th>
<th>Training time</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>Prec</td>
<td>Rec</td>
<td>F1</td>
<td>Prec</td>
</tr>
<tr>
<td>ConvAttention (Allamanis et al., 2016)</td>
<td>50.25</td>
<td>24.62</td>
<td>33.05</td>
<td>60.82</td>
</tr>
<tr>
<td>Paths+CRFs (Alon et al., 2018b)</td>
<td>8.39</td>
<td>5.63</td>
<td>6.74</td>
<td>32.56</td>
</tr>
<tr>
<td>code2vec (Alon et al., 2018a)</td>
<td>18.51</td>
<td>18.74</td>
<td>18.62</td>
<td>38.12</td>
</tr>
<tr>
<td>2-layer BiLSTM (full tokens)</td>
<td>32.40</td>
<td>20.40</td>
<td>25.03</td>
<td>48.37</td>
</tr>
<tr>
<td>2-layer BiLSTM (split tokens)</td>
<td>42.63</td>
<td>29.97</td>
<td>35.20</td>
<td>55.15</td>
</tr>
<tr>
<td>Transformer (Vaswani et al., 2017) (split tokens)</td>
<td>38.13</td>
<td>26.70</td>
<td>31.41</td>
<td>50.11</td>
</tr>
<tr>
<td>code2seq</td>
<td><strong>50.64</strong></td>
<td><strong>37.40</strong></td>
<td><strong>43.02</strong></td>
<td><strong>61.24</strong></td>
</tr>
</tbody>
</table>

**F1 for the code summarization task (Java)**

- ConvAttention
- Paths+CRFs
- code2vec
- BiLSTM (full tokens)
- BiLSTM (split tokens)
- Transformer (split tokens)
- code2seq
Example 1

```java
public boolean containsIgnoreCase(Set<String> set, String value) {
    for (String entry : set) {
        if (entry.equalsIgnoreCase(value))
            return true;
    }
    return false;
}
```
Example 1

```java
public boolean contains(Set<String> set, String value) {
    for (String entry : set) {
        if (entry.equalsIgnoreCase(value))
            return true;
    }
    return false;
}
```
Example 1

```java
public boolean containsIgnore(Set<String> set, String value) {
    for (String entry : set) {
        if (entry.equalsIgnoreCase(value))
            return true;
    }
    return false;
}
```
Example 1

```java
public boolean _____(Set<String> set,
                     String value) {
    for (String entry : set) {
        if (entry.equalsIgnoreCase(value))
            return true;
    }
    return false;
}
```

contains **ignore case**
Example 2

```csharp
TreeView myTreeView = new TreeView();
myTreeView.Nodes.Clear();
foreach (string parentText in xml.parent)
{
    TreeNode parent = new TreeNode();
    parent.Text = parentText;
    myTreeView.Nodes.Add(treeNodeDivisions);
    foreach (string childText in xml.child)
    {
        TreeNode child = new TreeNode();
        child.Text = childText;
        parent.Nodes.Add(child);
    }
}
```

**add a child node to a treeview in c#**

**how to get all child nodes in treeview**
TreeView myTreeView = new TreeView();
myTreeView.Nodes.Clear();
foreach (string parentText in xml.parent)
{
    TreeNode parent = new TreeNode();
    parent.Text = parentText;
    myTreeView.Nodes.Add(parentNodesDivisions);
    foreach (string childText in xml.child)
    {
        TreeNode child = new TreeNode();
        child.Text = childText;
        parent.Nodes.Add(child);
    }
}
TreeView myTreeView = new TreeView();
myTreeView.Nodes.Clear();
foreach (string parentText in xml.parent)
{
    TreeNode parent = new TreeNode();
    parent.Text = parentText;
    myTreeView.Nodes.Add(treeNodeDivisions);
    foreach (string childText in xml.child)
    {
        TreeNode child = new TreeNode();
        child.Text = childText;
        parent.Nodes.Add(child);
    }
}
add a child
TreeView myTreeView = new TreeView();
myTreeView.Nodes.Clear();
foreach (string parentText in xml.parent)
{
    TreeNode parent = new TreeNode();
    parent.Text = parentText;
    myTreeView.Nodes.Add(treeNodeDivisions);
    foreach (string childText in xml.child)
    {
        TreeNode child = new TreeNode();
        child.Text = childText;
        parent.Nodes.Add(child);
    }
}
add a child node to
add a child node to a
add a child node to a treeview

```csharp
TreeView myTreeView = new TreeView();
myTreeView.Nodes.Clear();
foreach (string parentText in xml.parent)
{
    TreeNode parent = new TreeNode();
    parent.Text = parentText;
    myTreeView.Nodes.Add(parent);
    foreach (string childText in xml.child)
    {
        TreeNode child = new TreeNode();
        child.Text = childText;
        parent.Nodes.Add(child);
    }
}
```
Example 3

```csharp
void Main()
{
    string text = File.ReadAllText(@"T:\File1.txt");
    int num = 0;
    text = (Regex.Replace(text, "map", delegate (Match m) {
               return "map" + num++;
            }));
    File.WriteAllText(@"T:\File1.txt", text);
}
```

Code2seq: replace a string in a text file

CodeNN: how to read a text file in c#
Example 4

```csharp
var excel = new ExcelQueryFactory("excelFileName");
var firstRow = excel.Worksheet().First();
var companyName = firstRow["CompanyName"];```

**Code2seq:**  
get the value of a column in excel using c#

**CodeNN:**  
how to get the value of an xml file in c#
Example 5

```csharp
static void Main(string[] args)
{
    // Create an instance of Bytescout.PDFRenderer.RasterRenderer object and register it.
    RasterRenderer renderer = new RasterRenderer();
    renderer.RegistrationName = "demo";
    renderer.RegistrationKey = "demo";
    // Load PDF document.
    renderer.LoadDocumentFromFile("multipage.pdf");
    for (int i = 0; i < renderer.GetPageCount(); i++)
    {
        // Render first page of the document to BMP image file.
        renderer.RenderPageToFile(i, RasterOutputFormat.BMP, "image" + i + ".bmp");
    }
    // Open the first output file in default image viewer.
    System.Diagnostics.Process.Start("image0.bmp");
}
```

**Code2seq:**
get the image from a pdf file in c#

**CodeNN:**
how to get the value of an array in c#
RNNs are awesome!

- Using LSTMs/GRUs to capture **regularity of code in programs**
- LSTMs/GRUs work well, but sometimes **surprise us**

- **What do** they actually learn?
  - Important to understand, especially when things go wrong
  - E.g., misclassification – can we provide examples that improve the net?

- **What can** they actually learn?
What has a network actually learned?
What has a network learned?

**valid email addresses**
40,000 training samples
2,000 test samples

Training
(100% accuracy on train, reached 100% also on test)

RNN

But has it really learned to recognize valid email addresses?
What has a network learned?

- **valid email addresses**
  - 40,000 training samples
  - 2,000 test samples

Training
(100% accuracy on train, reached 100% also on test)

RNN

- abc@sc.net
- blbl@df.com
- sf.se@sdf.co.uk
- ..@.co.
- dasd@@.vim
- b.net

- 25.net
- 5x.nem
- 2hs.net
How can we find these flaws?

valid email addresses
40,000 training samples
2,000 test samples

Training
(100% accuracy on train, reached 100% also on test)

RNN

- abc@sc.net
- blbl@df.com
- sf.se@sdf.co.uk
- ..@.co.
- dasd@@.vim
- b.net
- 25.net
- 5x.nem
- 2hs.net
What do RNNs actually learn?

• Novel extraction algorithm using abstraction and exact learning to interpret behaviour of RNNs
  • Dynamic abstraction guided by interaction with exact learning algorithm, L*
  • First application of exact learning to a trained RNN

• Creates accurate and concise automata for trained RNNs

• Quickly discovers adversarial inputs for seemingly perfect RNNs (this happens frustratingly often...!)
What can RNNs actually learn?

**Proof requires infinite precision**
"push 0 into stack": \( g = g/4 + 1/4 \)
this allows pushing 15 zeros when using 32 bit floating point.

**Construction requires complex integration of carefully crafted components**
can this really be reached by gradient methods?

**Construction requires extra processing time at the end of the sequence**
we use "real time" RNNs in practice

**Classical result: RNNs are Turing complete**
(a) $a^n b^n$-LSTM on $a^{1000} b^{1000}$

(b) $a^n b^n c^n$-LSTM on $a^{100} b^{100} c^{100}$

(c) $a^n b^n$-GRU on $a^{1000} b^{1000}$

(d) $a^n b^n c^n$-GRU on $a^{100} b^{100} c^{100}$
LSTM equations

\[
\begin{align*}
    f_t &= \sigma(W^f x_t + U^f h_{t-1} + b^f) \\
    i_t &= \sigma(W^i x_t + U^i h_{t-1} + b^i) \\
    o_t &= \sigma(W^o x_t + U^o h_{t-1} + b^o) \\
    \tilde{c}_t &= \tanh(W^c x_t + U^c h_{t-1} + b^c) \\
    c_t &= f_t \odot c_{t-1} + i_t \odot \tilde{c}_t \\
    h_t &= o_t \odot g(c_t)
\end{align*}
\]

Can make 1 via sigmoid

Exposes counter value
GRU Equations

\[ z_t = \sigma(W^z x_t + U^z h_{t-1} + b^z) \]

\[ r_t = \sigma(W^r x_t + U^r h_{t-1} + b^r) \]

\[ \tilde{h}_t = \tanh(W^h x_t + U^h (r_t \odot h_{t-1}) + b^h) \]

\[ h_t = (z_t \odot h_{t-1} + (1 - z_t) \odot \tilde{h}_t) \]
In the paper [ACL’18]

- RNNs as simplified k-counter machines
- Results comparing different architectures
  - SRNN/IRNN
  - GRU/LSTM
- Proof that SRNN/GRUs cannot implement a binary counter
Summary

code2vec
A (somewhat) interpretable model for predicting program properties

Extraction from RNN to DFA

What can RNNs learn?

(a) boolean
Oject (target) {
    for (Object elem: this.elements) {
        if (elem.equals(target)) {
            return true;
        }
    }
    return false;
}

(b) a^b^c^d^-LSTM on a^{1000}_{1000}

(c) a^b^c^d^-GRU on a^{1000}_{1000}

(d) a^b^c^d^-GRU on a^{1000}_{1000}

Prediction contains matches equals containsExact containsAll