Code Representation for Neural Networks and Applications

Tongyu Lu

SiDNN – 10.05.2022
Outline

Relevant Tasks

AST Code Representations

AST-based Models

Possible Research Directions
# Relevant Tasks

<table>
<thead>
<tr>
<th>Task Category</th>
<th>Input</th>
<th>Output</th>
</tr>
</thead>
<tbody>
<tr>
<td>Explanation Tasks</td>
<td>Code snippet</td>
<td>Natural language sequence</td>
</tr>
<tr>
<td>(code captioning, code summary)</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

**Code captioning in C#**:

```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 [1]** a **string [2]** in a **text [3]** file [3]

[Alon et al. 2018]

```csharp
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;
}
```

**Predictions**

- reverseArray: 77.34%
- reverse: 18.18%
- subArray: 1.45%
- copyArray: 0.74%
### Relevant Tasks

<table>
<thead>
<tr>
<th>Task Category</th>
<th>Input</th>
<th>Output</th>
</tr>
</thead>
<tbody>
<tr>
<td>Explanation Tasks</td>
<td>Code snippet</td>
<td>Natural language sequence</td>
</tr>
<tr>
<td>Explanation (code captioning, code summary)</td>
<td>Code snippet</td>
<td>Natural language sequence</td>
</tr>
<tr>
<td>Information Retrieval Tasks</td>
<td>Query String (e.g., key-word-to-find, code summary)</td>
<td>Relevant code (e.g., relevant identifiers, relevant code snippets)</td>
</tr>
</tbody>
</table>

---


<table>
<thead>
<tr>
<th>A</th>
<th>≈B</th>
</tr>
</thead>
<tbody>
<tr>
<td>size</td>
<td>getSize, length, getCount, getLength</td>
</tr>
<tr>
<td>active</td>
<td>isActive, setActive, setIsActive, enabled</td>
</tr>
<tr>
<td>done</td>
<td>end, stop, terminate</td>
</tr>
<tr>
<td>toJSON</td>
<td>serialize, toJSONString, getJson, asJson, execute, call, init, start</td>
</tr>
</tbody>
</table>

**Swap two elements in the list**

(a) Query $q$

1: void swapElementInList(List<Integer> list, int i, int j) {
2:     int element = list.get(i);
3:     list.set(i, list.get(j));
4:     list.set(j, element);
5: }

(b) Code Snippet $s_1$

1: void swapElementInList(List<Integer> list, int i, int j) {
2:     Collections.swap(list, i, j);
3: }

(c) Code Snippet $s_2$
# Relevant Tasks

<table>
<thead>
<tr>
<th>Task Category</th>
<th>Input</th>
<th>Output</th>
</tr>
</thead>
<tbody>
<tr>
<td>Explanation Tasks</td>
<td>Code snippet</td>
<td>Natural language sequence</td>
</tr>
<tr>
<td>(code captioning, code summary)</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Information Retrieval Tasks</td>
<td>Query String (e.g., key-word-to-find, code summary)</td>
<td>Relevant code (e.g., relevant identifiers, relevant code snippets)</td>
</tr>
<tr>
<td>(identifier name search, code search)</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Generation Tasks</td>
<td>Code snippet (incomplete) or natural language</td>
<td>Code snippet (e.g., a single identifier, a code block)</td>
</tr>
<tr>
<td>(code completion, comment to code)</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

Figure from [Li et al. 2017](#)
**Code Representation**

**Central problem**

**Question:** how to feed code to neural networks?

Option 1: NLP approach

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

Option 2: code as syntactic parse tree

Option 3: extract features
Question: how to feed code to neural networks?

Option 1: NLP approach

Option 2: code as syntactic parse tree

Option 3: extract features from parse tree
Abstract Syntax Tree (AST): parse tree for program codes

Example:

```java
while (!done) {
    if (someCondition()) {
        done = true;
    }
}
```
**Code Representation**

Bag of AST path contexts

- How to feed parse tree to neural network?

**AST path** example:

```
While
  UnaryPrefix!
    SymbolRef
      done
      someCondition
        SymbolRef
          SymbolRef
            Constant
              true
```

The red-marked path

\[ p = \langle \text{done}, \ (\text{SymbolRef} \uparrow \text{UnaryPrefix!} \uparrow \text{While} \downarrow \text{If} \downarrow \text{Assign=} \downarrow \text{Constant}), \text{true} \rangle \]
How to feed parse tree to neural network?

Proposed in code2vec [Alon et al. 2018].
**Code Representation**

**Bag of AST path contexts**

**Embedding for Bag of AST path contexts:**

Basic idea: maintain 2 embedding vocabularies: $V_{\text{value}}, V_{\text{path}}$

$$C = \{ (\text{done}, p_1, \text{someCondition}), (\text{done}, p_2, \text{done}), (\text{done}, p_3, \text{true}), (\text{someCondition}, p_4, \text{done}), (\text{someCondition}, p_5, \text{true}), (\text{done}, p_6, \text{true}) \}$$

$$\text{Emb}(⟨x_s, p, x_t⟩) = [V_{\text{value}}(x_s), V_{\text{path}}(p), V_{\text{value}}(x_t)]$$

Proposed as code2vec [Alon et al. 2018], further used in code2seq [Alon et al. 2019].

**Afraid of large vocabulary size?**

- Tokenize (e.g., `list_of_hash = [list, of, hash]`)
- Use RNN encoder for paths [Alon et al. 2018]
Code Representation

AST as sequence of (non-terminal, terminal) pairs

- Another idea to feed parse tree to neural network

What is the benefit?
Tokenized AST is a suitable representation for code completion [Li et al. 2017]
- AST is a graph!

**Question:** why not feed AST directly to GNN?

Reasoning of [Wang et al. 2021]: “in original AST, sequential information is missing”
Code Representation

Level 1: natural-language-like representations

Level 2: AST (syntax-level representation)

Level 3: extracted features (from AST)

Bag of AST paths,
Sequence of AST nodes (flattened AST),
AST graph

Summary

Bag of AST paths,
Sequence of AST nodes (flattened AST),
AST graph
Models

Utilizing AST Paths

Figure adapted from code2seq [Alon et al. 2019]

<table>
<thead>
<tr>
<th>Task Category</th>
<th>Input</th>
<th>Output</th>
</tr>
</thead>
<tbody>
<tr>
<td>Explanation Tasks</td>
<td>Code snippet</td>
<td>Natural language sequence</td>
</tr>
<tr>
<td>Information Retrieval Tasks</td>
<td>Query String</td>
<td>Relevant code</td>
</tr>
<tr>
<td>Code completion</td>
<td>Code snippet</td>
<td>Code snippet</td>
</tr>
</tbody>
</table>
Models

Utilizing AST Paths – researches by Alon et al.

Figure adapted from code2seq [Alon et al. 2019]

\[ c_t = \sum_{i} \alpha_i z_i \]

<table>
<thead>
<tr>
<th>Task Category</th>
<th>Input</th>
<th>Output</th>
</tr>
</thead>
<tbody>
<tr>
<td>Explanation Tasks</td>
<td>Code snippet</td>
<td>Natural language sequence</td>
</tr>
<tr>
<td>Information Retrieval Tasks</td>
<td>Query String</td>
<td>Relevant code</td>
</tr>
<tr>
<td>Code completion</td>
<td>Code snippet</td>
<td>Code snippet</td>
</tr>
</tbody>
</table>


**Models**

Utilizing AST Paths – researches by Alon et al.

\[ \alpha_i \]

**Figure adapted from code2vec [Alon et al. 2018]**

\( \tilde{c}_i \) attention with what?

A global vector \( a \) maintained as a parameter:

\[ \alpha_i = \text{softmax}(\tilde{c}_i^T a) \]

<table>
<thead>
<tr>
<th>Task Category</th>
<th>Input</th>
<th>Output</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Explanation Tasks</strong></td>
<td>Code snippet</td>
<td>Natural language sequence</td>
</tr>
<tr>
<td></td>
<td></td>
<td>A single word</td>
</tr>
<tr>
<td><strong>Information Retrieval Tasks</strong></td>
<td>Query String</td>
<td>Relevant code</td>
</tr>
<tr>
<td><strong>Code completion</strong></td>
<td>Code snippet</td>
<td>Code snippet</td>
</tr>
</tbody>
</table>
## Models

Utilizing AST Paths – researches by Alon et al.

**DEMO:** [https://code2seq.org/](https://code2seq.org/)

**DEMO**

[https://code2vec.org/](https://code2vec.org/) -> “most similar”, “analogy”

<table>
<thead>
<tr>
<th>Task Category</th>
<th>Input</th>
<th>Output</th>
</tr>
</thead>
<tbody>
<tr>
<td>Explanation Tasks</td>
<td>Code snippet</td>
<td>Natural language sequence</td>
</tr>
<tr>
<td>Information Retrieval Tasks</td>
<td>Query String</td>
<td>Relevant code</td>
</tr>
<tr>
<td>Code completion</td>
<td>Code snippet</td>
<td>Code snippet</td>
</tr>
</tbody>
</table>
### How to?

<table>
<thead>
<tr>
<th>Task Category</th>
<th>Input</th>
<th>Output</th>
</tr>
</thead>
<tbody>
<tr>
<td>Explanation Tasks</td>
<td>Code snippet</td>
<td>Natural language sequence</td>
</tr>
<tr>
<td>Information Retrieval Tasks</td>
<td>Query String</td>
<td>Relevant code</td>
</tr>
<tr>
<td>Code completion</td>
<td>Code snippet</td>
<td>Code snippet</td>
</tr>
</tbody>
</table>
Models

Single-token Code Completion Utilizing AST Token Sequences

Recall: converting code into AST token sequence

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

```
[(While,-),
 (UnaryPrefix!,-),
 (SymbolRef,done),
 (If,-),
 (Call,-),
 (SymbolRef, someCondition),
 (Assign=,-),
 (SymbolRef, done),
 (Constant, ?)]
```

Single-token code completion:
Predict the last token (which is exactly the end of DFS).
Models

Basic model: LSTM (with attention)

\[ c_t : \text{weighted sum with attention scores} \]
\[ p_t = h_{t-1} \in \mathbb{R}^d \]

Question: what if the desired prediction is not in the vocabulary?
Models

**Pointer Network**

2015, Vinyals et al., Pointer Network,
https://arxiv.org/abs/1506.03134v2

- Vanilla Seq2seq
- Pointer network
Models

Single-token code completion utilizing AST Token Sequences

\[
\begin{align*}
\text{Output distribution: } y_t &= [s_tw_t; (1 - s_t)l_t] \in \mathbb{R}^{V + L} \\
\text{Vocabulary candidates} \\
\text{Out-of-vocabulary (OoV) candidates} \\
s_t &= \sigma(\text{FC}([h_t, l_t])) \in [0,1] \\
\text{RNN distribution: } w_t &= \text{softmax}(\text{FC}([h_t, l_t, p_t])) \in \mathbb{R}^V
\end{align*}
\]

Figure from [Li et al. 2017]
Models

Single-token code completion utilizing AST Token Sequences

Example of OoV

```
class Operator(Employee):
    def __init__(self, name, employee_id):
        super(Operator, self).__init__(name, Rank.OPERATOR)
        self.employee_id = employee_id

    def _dispatch_call(self, call, employees):
        for employee in employees:
            employee.take_call(call)

    def record_path(self, base_name):
        return os.path.join(base_name, str(self.__?__))
```

Figure from [Li et al. 2017]
Models

Single-token code completion using GNN

Aggregation
1. Neighbor Graph Attention (NGAT)
2. Global Self-Attention (GSAT)
3. Parent-Child Attention (PCAT)
4. Residual connection

-> Get the final hidden state $H$

Summarization:

$$s = \text{weighted_pool}(H)$$

$$y^{(nt)} = \text{FC}(s),$$

$$y^{(t)} = \text{FC}(s)$$

Figure from [Wang et al. 2021]
Evaluating Large Language Models Trained on Code

Mark Chen*1  Jerry Tworek*1  Heewoo Jun*1  Qiming Yuan*1  Henrique Ponde de Oliveira Pinto*1  Jared Kaplan2  Harri Edwards3  Yuri Burda4  Nicholas Joseph5  Greg Brockman1  Alex Ray1  Raul Puri1  Gretchen Krueger1  Michael Petrov1  Heidy Khaaf1  Girish Sastry1  Pamela Mishkin1  Brooke Chan1  Scott Gray1  Nick Ryder1  Mikhail Pavlov1  Alethea Power1  Lukasz Kaiser1  Mohammad Bavarlan1  Clemens Winter1  Philippe Tillet1  Felipe Petroski Such1  Dave Cummings1  Matthias Plappert1  Fotios Chantatzis1  Elizabeth Barnes1  Ariel Herbert-Voss1  William Hehgen Guss1  Alex Nichol1  Alex Paino1  Nikolas Tezak1  Jie Tang1  Igor Babuschkin1  Suchi Balaji1  Shantanu Jain1  William Saunders1  Christopher Hesse1  Andrew N. Carr1  Jan Leike1  Josh Achiam1  Vedant Misra1  Evan Morikawa1  Alec Radford1  Matthew Knight1  Miles Brundage1  Mira Marati1  Katie Mayer1  Peter Wellner1  Bob McGrew1  Darío Amodei2  Sam McCandlish2  Ilya Sutskever1  Wojciech Zaremba1

https://copilot.github.com/
Want-to-knows:

- Representation of $\hat{w}_t$
  - Problems of predicting natural-language-level tokens?
  - Problems of predicting AST token pairs?

- OoV?
## Summary

Contents covered

- AST-based representations
- Code2Seq, Single-token code completion

<table>
<thead>
<tr>
<th>Work</th>
<th>Code Representation</th>
<th>Task</th>
<th>Model</th>
</tr>
</thead>
<tbody>
<tr>
<td>CCAG [Wang et al. 2021]</td>
<td>AST Token Sequence</td>
<td>Code completion</td>
<td>Pointer Mixture Network</td>
</tr>
<tr>
<td></td>
<td>AST Graph</td>
<td>Code completion</td>
<td>GNN</td>
</tr>
</tbody>
</table>

Tensors are universal
Possible Research Directions

A tentative list of relevant topics for research

GNN-related open questions and “combination of techniques” (which is not done yet)
- AST vs. flattened AST graph: does “sequential information” really matter?
- OoV (graph pointer neural network)
- More GNN architectures

Code-block completion
- How did Codex achieve this?
- Is it possible to generate code in a natural-language-like manner?
- How to generate AST using neural network?

More application scenarios
- e.g., code maintenance: given description (e.g., “plot the output”) and modify the original code
- ...
References


Back-up Contents
Code Representation  

Variable flow

Proposed in GraphCodeBERT [Guo et al. 2021]

Source code

def max(a, b):
    x=0
    if b>a:
        x=b
    else:
        x=a
    return x

Figures adapted from [Guo et al. 2021], GraphCodeBERT

How does low-resolution feature help?
Code Search


```java
// calculate the sum of an int array
public int calArraySum(int[] array) {
    int sum = 0;
    int i = 0;
    for (; i < array.length; i++) {
        sum = sum + array[i];
    }
    return sum;
}
```

(a) Code Snippet $S_a$

0: push int constant 0.
1: store int 0 into local variable sum/result.
2: push int constant 0.
3: store int 0 into local variable i/index.
4: load int value from local variable i/index.
5: load reference array/array from local variable array/array.
6: get length of array array/array.
7: if and only if int value is greater or equal to int length then go to 22
10: load int value_1 from local variable sum/result.
11: load reference array/array from local variable array/array.
12: load int value_2 from local variable i/index.
13: load int value_3 from array/array[value_2].
14: int result is int value_1 add int value_3; push result into value_4.
15: store int value_4 into local variable sum/result.
16: increment local variable i/index by constant 1.
19: goto 4.
22: load int value_5 from local variable sum/result.
23: return int value_5 from method.
Filling in the blank given a partial AST.

The output space in each generation step is determined by the previous token.

Generation ends when sampling EOS_token or EOS_node.

```java
public static Path[] stat2Paths(
    FileStatus[] stats) {
    if (stats == null) return null;
    Path[] ret = new Path[stats.length];
    for (int i = 0; i < stats.length; ++i){
        ret[i] = _
    }
    return ret;
}
```

```java
public static string Camelize(
    this string input)
{
    var word = input.Pascalize();
    return word.Length > 0 ?
        word.ToLower()
        + word.Substring(1)
    : word;
}
```
Datasets

For code completion:
JS (JS50K etc.), PY (PY50K etc.) Datasets: https://www.sri.inf.ethz.ch/research/plml

For code summary:
Java (Java Large etc.) Datasets: https://groups.inf.ed.ac.uk/cup/codeattention/
CodeNN C# dataset: https://github.com/sriniiyer/codnn/

For code search:
Model Performances

Single-token code completion
Metric: accuracy

<table>
<thead>
<tr>
<th></th>
<th>JS1k</th>
<th>JS10k</th>
<th>JS50k</th>
<th>PY1k</th>
<th>PY10k</th>
<th>PY50k</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>value</td>
<td>type</td>
<td>value</td>
<td>type</td>
<td>value</td>
<td>type</td>
</tr>
<tr>
<td>VanillaLSTM</td>
<td>53.19%</td>
<td>69.52%</td>
<td>58.04%</td>
<td>71.16%</td>
<td>59.70%</td>
<td>72.08%</td>
</tr>
<tr>
<td>ParentLSTM</td>
<td>56.45%</td>
<td>71.99%</td>
<td>61.54%</td>
<td>73.46%</td>
<td>63.39%</td>
<td>74.24%</td>
</tr>
<tr>
<td>PointerMixtureNet</td>
<td>56.49%</td>
<td>71.95%</td>
<td>62.33%</td>
<td>74.28%</td>
<td>64.14%</td>
<td>76.01%</td>
</tr>
<tr>
<td>Transformer</td>
<td>58.40%</td>
<td>73.29%</td>
<td>63.93%</td>
<td>74.78%</td>
<td>65.31%</td>
<td>75.89%</td>
</tr>
<tr>
<td>Transformer-XL</td>
<td>59.23%</td>
<td>72.11%</td>
<td>62.82%</td>
<td>74.09%</td>
<td>66.41%</td>
<td>76.23%</td>
</tr>
<tr>
<td>CCAG</td>
<td>62.79%</td>
<td>75.72%</td>
<td>66.69%</td>
<td>78.55%</td>
<td>68.19%</td>
<td>80.14%</td>
</tr>
</tbody>
</table>

Code Summary

<table>
<thead>
<tr>
<th>Model</th>
<th>Full Test Set (413915 methods)</th>
<th>Precision</th>
<th>Recall</th>
<th>F1</th>
</tr>
</thead>
<tbody>
<tr>
<td>CNN+Attention [Allamanis et al. 2016]</td>
<td>-</td>
<td>-</td>
<td>-</td>
<td>-</td>
</tr>
<tr>
<td>LSTM+Attention [Iyer et al. 2016]</td>
<td>33.7</td>
<td>22.0</td>
<td>26.6</td>
<td></td>
</tr>
<tr>
<td>Paths+CRFs [Alon et al. 2018]</td>
<td>53.6</td>
<td>46.6</td>
<td>49.9</td>
<td></td>
</tr>
<tr>
<td>PathAttention (this work)</td>
<td>63.1</td>
<td>54.4</td>
<td>58.4</td>
<td></td>
</tr>
</tbody>
</table>

On Java dataset