Learning from Source Code Identifiers

Miltos Allamanis
Learning to Name Source Code

A name reflects important aspects of code functionality.

Learning to name source code is a first step in understanding code through machine learning.
ForkJoinTask<?> job;
if (task instanceof ForkJoinTask<?>)) // avoid re-wrap
    job = (ForkJoinTask<?>) task;
else
    job = new ForkJoinTask.AdaptedRunnableAction(task);
    externalPush(job);

Suggestions
1. job (30%)
2. task (20%)
3. tsk (15%)
A Machine Learning Model of Names

\[ P(t | c) = \frac{\exp(s_\theta(t, c))}{\sum_{t'} \exp(s_\theta(t', c))} \]

Embedding Identifiers

$q_t \in \mathbb{R}^D$ are “embeddings” :: model parameters

v(king) - v(man) + v(women) = v(queen)

The Distributional Hypothesis in Linguistics

The ????????? is walking

“You shall know a word by the company it keeps”.

–John Rupert Firth, 1957
Embedding Identifiers

\[ s_\theta(t, c) = \hat{r}_{\text{context}}^\top q_t + b_t \]

\[ P(t|c) = \frac{\exp(s_\theta(t, c))}{\sum_{t'} \exp(s_\theta(t', c))} \]

Neural Context Models of Source Code

Variable: \texttt{isDone}

\[
s_\theta(.) = \hat{r}_{\text{context}}^T \mathbf{q}_{\text{isDone}} + b_{\text{isDone}}
\]
Neural Context Models of Source Code

Variable: $\text{isDone}$

$s_\theta(.) = \mathbf{r}_{\text{context}}^T \mathbf{q}_{\text{isDone}} + b_{\text{isDone}}$
Neural Context Models of Source Code

Variable: $\text{isDone}$

Features:
- boolean, in: MethodBody, final

$$s_\theta(.) = \hat{r}_{\text{context}}^T q_{\text{isDone}} + b_{\text{isDone}}$$
Neural Context Models of Source Code

Variable: \( \text{isDone} \)

Features:
- boolean, in:MethodBody, final

Contexts:
- while(!\( \text{isDone} \)) {

\[ s_\theta(.) = \hat{r}_{\text{context}}^T q_{\text{isDone}} + b_{\text{isDone}} \]
Neural Context Models of Source Code

Variable: \( \text{isDone} \)

Features:
- boolean, in:MethodBody, final
- \( \mathbf{r}_{\text{boolean}} + \mathbf{r}_{\text{in:MethodBody}} + \mathbf{r}_{\text{final}} \)

Contexts:
- while \((! \text{isDone})\) {
  - \( C_{-2} \mathbf{r} + C_{-1} \mathbf{r} + C_{1} \mathbf{r} + C_{2} \mathbf{r} \)

\( s_{\theta}(.) = \mathbf{r}_{\text{context}}^T \text{isDone} + b_{\text{isDone}} \)
Neural Context Models of Source Code

Variable: \( \text{isDone} \)

Features:

\begin{align*}
\text{boolean, } \text{in:MethodBody, final} & \\
\mathbf{r}_{\text{boolean}} + \mathbf{r}_{\text{in:MethodBody}} + \mathbf{r}_{\text{final}} & \\
\hat{r}_{\text{context}} = \sum_{f \in F_{\text{te}}} r_f + \sum_{\forall k: K \geq |k| > 0} C_k r_{t+i+k} & \\
\end{align*}

Contexts:

while \((\neg \text{isDone})\) {
\begin{align*}
C_{-2} r_2 + C_{-1} r_1 + C_1 r + C_2 r & \\
\end{align*}

\[ s\theta(.) = \hat{r}_{\text{context}}^T \text{isDone} + b_{\text{isDone}} \]
Neural Context Models of Source Code

Variable: \text{isDone}

Features:
- boolean, in:MethodBody, final

\[ r_{\text{context}} = \sum_{f \in F_{te}} r_f + \sum_{\forall k: K \geq |k| > 0} C_k r_{t_i+k} \]

Contexts:
- while ( ! \text{isDone} ) {
  \[ C_{-2} r - C_{-1} r_i + C_1 r + C_2 r \]
}
Neural Context Models of Source Code

Global Information
Neural Context Models of Source Code
Neural Context Models of Source Code

Variable: \textbf{isDone}

Features:
- boolean, \textbf{in:MethodBody}, final

\[ \mathbf{r}_{context} = \sum_{f \in F_{tc}} \mathbf{r}_f + \sum_{\forall k: K \geq |k| > 0} C_k \mathbf{r}_{t_i+k} \]

Contexts:
- while (!isDone) {
  \[ C_{-2} r_{...} + C_{-1} r_1 + C_1 r_1 + C_2 r_1 \]
Embedding Identifiers

\[ P(t|c) = \frac{\exp(s_\theta(t, c))}{\sum_{t'} \exp(s_\theta(t', c))} \]

Neologisms

neologism

[ˌnɪəˈlɒdʒɪz(ə)m]  \( \text{NOUN} \)

a newly coined word or expression.

synonyms: new word · new expression · new term · new phrase · coinage · More

Powered by OxfordDictionaries · © Oxford University Press
Subtoken Context Models of Code

Sequentially predict each subtoken given the context and the previous subtokens

\[ P(t_i|t_{i-1}, t_{i-2}, \text{context}) \]
Training Data (project) → Train Neural Network → Embeddings

- Suggest Names on Test Data

Graph shows the relationship between Suggestion Frequency and Suggestion Pt.
Embedding Visualization

T-SNE

http://groups.inf.ed.ac.uk/cup/naturalize
## Embeddings - Closely Related Terms

<table>
<thead>
<tr>
<th>Feature Model</th>
<th>Subtoken</th>
</tr>
</thead>
<tbody>
<tr>
<td>camera - cam</td>
<td>6 - 5</td>
</tr>
<tr>
<td>padBottom - padLeft</td>
<td>Height - Width</td>
</tr>
<tr>
<td>dataOut - dataIn</td>
<td>swig - class</td>
</tr>
<tr>
<td>localAnchorA - localAnchorB</td>
<td>Min - Max</td>
</tr>
<tr>
<td>bodyA - bodyB</td>
<td>shape - collision</td>
</tr>
<tr>
<td>framebuffers - buffers</td>
<td>Left - Right</td>
</tr>
<tr>
<td>worldWidth - worldHeight</td>
<td>camera - cam</td>
</tr>
<tr>
<td>padRight - padLeft</td>
<td>TOUCH - KEY</td>
</tr>
<tr>
<td>jarg7 - jarg6_</td>
<td>end - start</td>
</tr>
<tr>
<td>spriteBatch - batch</td>
<td>loc - location</td>
</tr>
</tbody>
</table>
Setters vs Getters in bigbluebutton
Singular vs Plural in libgdx
Learning to Name Methods
```java
private void () {
    String vertexShader = "literal_1";
    String fragmentShader = "literal_2";
    shader = new ShaderProgram(vertexShader,
                                fragmentShader);
    if (shader.isCompiled() == false)
        throw new IllegalArgumentException("literal_3" + shader.getLog());
}
```
Method Naming Problem

```java
private void * ( ) {
    String vertexShader = "literal_1";
    String fragmentShader = "literal_2";
    shader = new ShaderProgram(vertexShader,
    fragmentShader);
    if(shader.isCompiled() == false)
        throw new IllegalArgumentException(
            "literal_3" + shader.getLog());
}
```

Names describe what it *does* not what it *is*

Models need to be “non-local”
Method Naming Problem

```java
private void createShader() {
    String vertexShader = "literal_1";
    String fragmentShader = "literal_2";
    shader = new ShaderProgram(vertexShader, fragmentShader);
    if (shader.isCompiled() == false)
        throw new IllegalArgumentException(
            "literal_3" + shader.getLog());
}
```

Suggestions:
- create
- create?
- init
- createShader
Method Naming Problem

```java
private void createDefaultShader() {
    String vertexShader = "literal_1";
    String fragmentShader = "literal_2";
    shader = new ShaderProgram(vertexShader,
                                fragmentShader);
    if (shader.isCompiled() == false)
        throw new IllegalArgumentException(
            "literal_3" + shader.getLog());
}
```

Suggestions:
• create • create?UNK? • init • createShader
Convolution

From deeplearningbook.org
1D Convolution
Attention Mechanisms in Deep Learning
The RNN gives an attention distribution which describe how we spread out the amount we care about different memory positions.

The read result is a weighted sum.

\[ r \leftarrow \sum_i a_i M_i \]
Convolutional Attention Network for Naming Methods

Joint work with Hao Peng, Charles Sutton
ICML 2016
Neural Attention Models for Method Names

<s> { return payload != null ; } </s>

Method name starts with **is**

isDragging()
Architecture

(Subtoken) Summary

<table>
<thead>
<tr>
<th>min</th>
<th>Run</th>
<th>Length</th>
</tr>
</thead>
<tbody>
<tr>
<td>( m_1 )</td>
<td>( m_2 )</td>
<td>( m_3 )</td>
</tr>
</tbody>
</table>

An RNN to predict summary subtokens

\[ P(m_i | m_0 \ldots m_{i-1}, \text{code}) \]

if (DEBUG) assert n >= 0;
int r = 0;
while (n >= MIN_MERGE) {
    r |= (n & 1);
    n >>= 1;
}
return n + r;

Code

Context-Dependent Convolutional Attention Features
synchronized(sGlThreadManager) {
    return mRenderMode;
}

Embedding
Code
Subtokens

Extracting Attention
Features
Computing Attention Features
Context-dependent

$E_{mt}$

$K_{l1}$

$K_{l2}$

ReLU

$h_{t-1}$

$k_1$

$k_2$

$L_{feat}$ attention features

Subtokens
**attention_features** (code tokens \( c \), context \( h_{t-1} \))

\[
C \leftarrow \text{LOOKUPANDPAD}(c, E)
\]

\[
L_1 \leftarrow \text{RELU}(\text{CONV1D}(C, K_{l1}))
\]

\[
L_2 \leftarrow \text{CONV1D}(L_1, K_{l2}) \odot h_{t-1}
\]

\[
L_{feat} \leftarrow L_2 / \|L_2\|_2
\]

**return** \( L_{feat} \)

---

Extracting Attention Features
GRU (current input $x_t$, previous state $h_{t-1}$)

$r_t \leftarrow \sigma(x_t W_{xr} + h_{t-1} W_{hr} + b_r)$
$u_t \leftarrow \sigma(x_t W_{xu} + h_{t-1} W_{hu} + b_u)$
$c_t \leftarrow \tanh(x_t W_{xc} + r_t \circ (h_{t-1} W_{hc}) + b_c)$
$h_t \leftarrow (1 - u_t) \circ h_{t-1} + u_t \circ c_t$

return $h_t$
Predicting the Next Method Subtoken

\[ \text{conv\_attention} \text{ (code } c, \text{ previous state } h_{t-1}) \]
\[ \alpha \leftarrow \text{attention\_weights} \left( L_{\text{feat}}, K_{\text{att}} \right) \]
\[ \hat{n} \leftarrow \sum_{i} \alpha_i E_{c_i} \]
\[ n \leftarrow \text{SOFTMAX}(E \hat{n}^\top + b) \]
return \( n \)
Computing Multiple Attention Weights
copy_attention (code $c$, previous state $h_{t-1}$)

$L_{feat} \leftarrow \text{attention_features} (c, h_{t-1})$

$\alpha_{att} \leftarrow \text{attention_weights} (L_{feat}, K_{att})$

$\alpha_{copy} \leftarrow \text{attention_weights} (L_{feat}, K_{copy})$

$\lambda \leftarrow \max (\sigma(\text{CONV1D}(L_{feat}, K_{\lambda})))$

$\hat{n} \leftarrow \sum_i \alpha_i E_{c_i}$

$n \leftarrow \text{SOFTMAX}(E \hat{n}^T + b)$

return $\lambda \text{POS2VOC}(\kappa, c) + (1 - \lambda) \text{TOMAP}(n, V)$
**copy_attention** (code $c$, previous state $h_{t-1}$)

- $L_{feat} \leftarrow \text{attention_features} (c, h_{t-1})$
- $\alpha_{att} \leftarrow \text{attention_weights} (L_{feat}, K_{att})$
- $\alpha_{copy} \leftarrow \text{attention_weights} (L_{feat}, K_{copy})$
- $\lambda \leftarrow \max(\sigma(\text{CONV1D}(L_{feat}, K_{\lambda})))$
- $\hat{n} \leftarrow \sum_i \alpha_i E_{c_i}$
- $n \leftarrow \text{SOFTMAX}(E \hat{n}^T + b)$
- return $\lambda \text{POS2VOC}(\kappa, c) + (1 - \lambda) \text{TO_MAP}(n, V)$

**meta-attention mechanism**
Architecture

(Subtoken) Summary

<table>
<thead>
<tr>
<th>min</th>
<th>Run</th>
<th>Length</th>
</tr>
</thead>
<tbody>
<tr>
<td>(m_1)</td>
<td>(m_2)</td>
<td>(m_3)</td>
</tr>
</tbody>
</table>

An RNN to predict summary subtokens

\[ P(m_i | m_0 \ldots m_{i-1}, \text{code}) \]

Code

```
if (DEBUG) assert n >= 0;
int r = 0;
while (n >= MIN_MERGE) {
    r |= (n & 1);
    n >>= 1;
}
return n + r;
```

Context-Dependent Convolutional Attention Features
<table>
<thead>
<tr>
<th>Project</th>
<th>Forks</th>
<th>Stars</th>
<th># Train Samples</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>cassandra</td>
<td>1,275</td>
<td>2,795</td>
<td>11,490</td>
<td>Distributed Database</td>
</tr>
<tr>
<td>elasticsearch</td>
<td>5,726</td>
<td>16,923</td>
<td>15,330</td>
<td>REST Search Engine</td>
</tr>
<tr>
<td>gradle</td>
<td>1,407</td>
<td>3,581</td>
<td>11,860</td>
<td>Build System</td>
</tr>
<tr>
<td>hadoop-common</td>
<td>926</td>
<td>610</td>
<td>25,864</td>
<td>Map-Reduce Framework</td>
</tr>
<tr>
<td>hibernate-orm</td>
<td>1,511</td>
<td>1,858</td>
<td>17,950</td>
<td>Object-Relational Mapping</td>
</tr>
<tr>
<td>intellij-community</td>
<td>1,355</td>
<td>3,119</td>
<td>236,365</td>
<td>IDE</td>
</tr>
<tr>
<td>liferay-portal</td>
<td>1,545</td>
<td>889</td>
<td>70,389</td>
<td>Portal Framework</td>
</tr>
<tr>
<td>presto</td>
<td>1,437</td>
<td>4,815</td>
<td>8,584</td>
<td>Distributed SQL query engine</td>
</tr>
<tr>
<td>spring-framework</td>
<td>7,559</td>
<td>9,160</td>
<td>23,372</td>
<td>Application Framework</td>
</tr>
<tr>
<td>wildfly</td>
<td>1,594</td>
<td>1,620</td>
<td>23,365</td>
<td>Application Server</td>
</tr>
</tbody>
</table>

**Dataset**  
http://groups.inf.ed.ac.uk/cup/codeattention/
<table>
<thead>
<tr>
<th></th>
<th>F1 (%)</th>
<th></th>
<th>Exact Match (%)</th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>Rank 1</td>
<td>Rank 5</td>
<td>Rank 1</td>
<td>Rank 5</td>
</tr>
<tr>
<td>tf-idf</td>
<td>40.0</td>
<td>52.1</td>
<td>24.3</td>
<td>29.3</td>
</tr>
<tr>
<td>Standard Attention</td>
<td>33.6</td>
<td>45.2</td>
<td>17.4</td>
<td>24.9</td>
</tr>
<tr>
<td>No Copy Network</td>
<td>43.6</td>
<td>57.7</td>
<td>20.6</td>
<td>29.8</td>
</tr>
<tr>
<td>Full Network</td>
<td>44.7</td>
<td>59.6</td>
<td>23.5</td>
<td>33.7</td>
</tr>
</tbody>
</table>

Bahdanau, Cho, and Bengio. “Neural machine translation by jointly learning to align and translate”. 2015
<table>
<thead>
<tr>
<th>Target Name</th>
<th>set ( m_1 )</th>
<th>use ( m_2 )</th>
<th>browser ( m_3 )</th>
<th>cache ( m_4 )</th>
<th>( \lambda )</th>
</tr>
</thead>
<tbody>
<tr>
<td>( m_1 ) set</td>
<td>( \alpha_{\text{att}} = \langle s \rangle { \text{this . use Browser Cache} = \text{use Browser Cache} ; } \langle / s \rangle )</td>
<td>( \alpha_{\text{copy}} = \langle s \rangle { \text{this . use Browser Cache} = \text{use Browser Cache} ; } \langle / s \rangle )</td>
<td></td>
<td></td>
<td>0.012</td>
</tr>
<tr>
<td>( m_2 ) use</td>
<td>( \alpha_{\text{att}} = \langle s \rangle { \text{this . use Browser Cache} = \text{use Browser Cache} ; } \langle / s \rangle )</td>
<td>( \alpha_{\text{copy}} = \langle s \rangle { \text{this . use Browser Cache} = \text{use Browser Cache} ; } \langle / s \rangle )</td>
<td></td>
<td></td>
<td>0.974</td>
</tr>
<tr>
<td>( m_3 ) browser</td>
<td>( \alpha_{\text{att}} = \langle s \rangle { \text{this . use Browser Cache} = \text{use Browser Cache} ; } \langle / s \rangle )</td>
<td>( \alpha_{\text{copy}} = \langle s \rangle { \text{this . use Browser Cache} = \text{use Browser Cache} ; } \langle / s \rangle )</td>
<td></td>
<td></td>
<td>0.969</td>
</tr>
<tr>
<td>( m_4 ) cache</td>
<td>( \alpha_{\text{att}} = \langle s \rangle { \text{this . use Browser Cache} = \text{use Browser Cache} ; } \langle / s \rangle )</td>
<td>( \alpha_{\text{copy}} = \langle s \rangle { \text{this . use Browser Cache} = \text{use Browser Cache} ; } \langle / s \rangle )</td>
<td></td>
<td></td>
<td>0.583</td>
</tr>
<tr>
<td>( m_5 ) END</td>
<td>( \alpha_{\text{att}} = \langle s \rangle { \text{this . use Browser Cache} = \text{use Browser Cache} ; } \langle / s \rangle )</td>
<td>( \alpha_{\text{copy}} = \langle s \rangle { \text{this . use Browser Cache} = \text{use Browser Cache} ; } \langle / s \rangle )</td>
<td></td>
<td></td>
<td>0.066</td>
</tr>
</tbody>
</table>

Attention Visualization

Data and Visualizations:
http://groups.inf.ed.ac.uk/cup/codeattention/
<table>
<thead>
<tr>
<th>Target</th>
<th>Attention Vectors</th>
<th>$\lambda$</th>
</tr>
</thead>
<tbody>
<tr>
<td>$m_1$</td>
<td>$\alpha = \langle s \rangle { \text{return (mFlags &amp; eBulletFlag) == eBulletFlag; } } \langle /s \rangle$</td>
<td>0.012</td>
</tr>
<tr>
<td></td>
<td>$\kappa = \langle s \rangle { \text{return (mFlags &amp; eBulletFlag) == eBulletFlag; } } \langle /s \rangle$</td>
<td></td>
</tr>
<tr>
<td>$m_2$</td>
<td>$\alpha = \langle s \rangle { \text{return (mFlags &amp; eBulletFlag) == eBulletFlag; } } \langle /s \rangle$</td>
<td>0.436</td>
</tr>
<tr>
<td></td>
<td>$\kappa = \langle s \rangle { \text{return (mFlags &amp; eBulletFlag) == eBulletFlag; } } \langle /s \rangle$</td>
<td></td>
</tr>
<tr>
<td>$m_3$</td>
<td>$\alpha = \langle s \rangle { \text{return (mFlags &amp; eBulletFlag) == eBulletFlag; } } \langle /s \rangle$</td>
<td>0.174</td>
</tr>
<tr>
<td></td>
<td>$\kappa = \langle s \rangle { \text{return (mFlags &amp; eBulletFlag) == eBulletFlag; } } \langle /s \rangle$</td>
<td></td>
</tr>
</tbody>
</table>
```java
void reverseRange(Object[] a, int lo, int hi)

hi--;  
while (lo < hi) {
    Object t = a[lo];
    a[lo++] = a[hi];
    a[hi--] = t;
}
```

**Suggestions**

- reverse, range (22%)
- reverse (13%)
- reverse, lo (4%)
- reverse, hi (3%)
```java
int createProgram()
{
    GL20 gl = Gdx.gl20;
    int program = gl.glCreateProgram();
    return program != 0 ? program : -1;
}
```

**Suggestions**
- create (18%)
- init (8%)
- render (5%)
- initiate (5%)
float getAspectRatio()

return (height == 0) ? Float.NaN : width / height;

Predictions
- get, UNK (9%)
- get, height (8.7%)
- get, width (6.5%)
- get (5.7%)
- get, size (4.2%)
```java
boolean shouldRender()
{
    try {
        return renderRequested || isContinuous;
    } finally {
        renderRequested = false;
    }
}

Predictions
• is, render (27%)
• is, continuous (11%)
• is, requested (8%)
• render, continuous (7%)
```
**This is your machine learning system?**

**Yup! You pour the data into this big pile of linear algebra, then collect the answers on the other side.**

**What if the answers are wrong?**

**Just stir the pile until they start looking right.**
Model Capacity (*what can the model learn?*)
- Overtrain on a small dataset
- Synthetic data

Optimization Issues (*can we make the model learn?*)
- Look at training curves
- Monitor gradient update ratios
- Hand-pick parameters for synthetic data

Other model “bugs” (*is the model doing what I want it to do?*)
- Generate samples from your model (if you can)
- Visualize learned representations (*e.g.* embeddings, nearest neighbors)
- Error analysis (examples where the model is failing, most “confident” errors)
- Simplify the problem/model
- Increase capacity, sweep hyperparameters (*e.g.* increase size of \( h \) in LSTM)

Practical (?) Tips on Debugging Machine Learning Models
Learning Python Code Suggestion with a Sparse Pointer Network.
Bhoopchand, Avishkar, et al.

Sub-token Language Models

Attention over symbol table

Some More Code Completion Ideas...
Machine Learning Models

- Statistical Code Migration
- Code Synthesis
- Code Defects
- Program Analysis
- Coding Conventions

Recommender Systems

Code Documentation

Code Summarization