Language-Agnostic Representation Learning of Source Code from Structure and Context

Daniel Zügner¹, Tobias Kirschstein¹, Michele Catasta², Jure Leskovec², Stephan Günnemann¹

¹Technical University of Munich
²Stanford University
def func(s):
    l = s.lower()
    if l in ('true', 't', '1'):
        return True
    if l in ('false', 'f', '0'):
        return False
    raise Exception("Unable to convert string '%s' to a boolean value" % s)
One of the hardest problems in Computer Science

```python
def bool_(s):
    l = s.lower()
    if l in ("true", "t", "1"):
        return True
    if l in ("false", "f", "0"):
        return False
    raise Exception(
        "Unable to convert string '%s' to a boolean value" % s
    )
```

3
One of the hardest problems in Computer Science

```python
def parse_bool(s):
    l = s.lower()
    if l in ('true', 't', '1'):
        return True
    if l in ('false', 'f', '0'):
        return False
    raise Exception('Unable to convert string %s to a boolean value' % s)
```
Machine Learning for Code

In **ML4Code**, the goal is to use ML to make a developer’s life easier, e.g., by

- **Suggesting** method or variable names
- Finding **similar or related** code snippets
- **Spotting** and/or repairing **bugs**
- Improving **the code** by optimizing or refactoring
- **Generating code** from natural language prompts
Structure and Context Representation

Structure and Context are complementary representations of a program. Most works leverage only one of them. We propose to combine them.
Transformers on Structured Data

(a) Transformer on Text

(b) Transformer on Graphs

ENCODER LAYER

Self-Attention

Absolute position encodings

enc(p_1) enc(p_2) enc(p_3) enc(p_4)

Transformers are very expressive

ENCODER LAYER

Self-Attention

Relational Attention

Input encodings

enc(r_{12}) enc(r_{13}) enc(r_{14})

No absolute positions in relational data

1 2 3 4
Transformers on Structured Data

(a) Transformer on Text

(b) Transformer on Graphs

(c) Code Transformer
Relative Distances on the AST

Instead of proprietary, programming-language-specific pre-processing, we only use language-agnostic features from the AST.
## Results: Code Summarization

<table>
<thead>
<tr>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>code2seq</td>
<td>35.79</td>
<td>24.85</td>
<td>29.34</td>
<td>30.18</td>
<td>19.88</td>
<td>23.97</td>
<td>23.23</td>
<td>10.31</td>
<td>14.28</td>
<td>52.30</td>
<td>43.43</td>
<td>47.45</td>
</tr>
<tr>
<td>GREAT</td>
<td>35.07</td>
<td>31.59</td>
<td>33.24</td>
<td>31.20</td>
<td>26.84</td>
<td>28.86</td>
<td>24.64</td>
<td>22.23</td>
<td>23.38</td>
<td>50.01</td>
<td>46.51</td>
<td>48.20</td>
</tr>
<tr>
<td>Code Transformer</td>
<td>36.40</td>
<td>33.66</td>
<td>34.97</td>
<td>35.06</td>
<td>29.61</td>
<td>32.11</td>
<td>31.42</td>
<td>24.46</td>
<td>27.50</td>
<td>55.51</td>
<td>48.05</td>
<td>51.34</td>
</tr>
<tr>
<td>Code Transformer (Multilanguage)</td>
<td>38.89</td>
<td>33.82</td>
<td>36.18</td>
<td>36.95</td>
<td>29.98</td>
<td>33.19</td>
<td>33.93</td>
<td>28.94</td>
<td>31.24</td>
<td>56.00</td>
<td>50.44</td>
<td>53.97</td>
</tr>
<tr>
<td>Code Transformer (Mult. + Pretrain)</td>
<td>39.67</td>
<td>35.29</td>
<td>37.35</td>
<td>37.06</td>
<td>31.94</td>
<td>34.31</td>
<td>35.19</td>
<td>29.36</td>
<td>32.01</td>
<td>57.73</td>
<td>51.89</td>
<td>54.65</td>
</tr>
</tbody>
</table>
Multilanguage Embeddings

We can map code snippets from **different languages** into a **shared embedding space**.

We can see that **similar methods** are mapped to regions close by in **embedding space**.
Summary

• We propose the Code Transformer for representation learning on code.

• Our language-agnostic design enables our model to jointly learn on multiple programming languages.

• Multilanguage training improves results on all individual languages, with strongest gains on low-resource languages.

• All model & pipeline code and pre-trained models are publicly available.

Project page: www.daml.in.tum.de/code-transformer
Code: https://github.com/danielzuegner/code-transformer
Demo: http://code-transformer.org