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

Daniel Zügner[1], Tobias Kirschstein[1], Michele Catasta[2],
Jure Leskovec[2], Stephan Günnemann[1]

[1] Technical University of Munich
[2] Stanford University

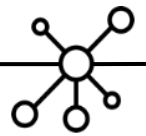
Stanford University

One of the hardest problems in Computer Science

```
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

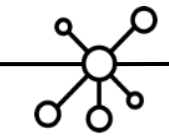
```
def ▌(s):
    l = s.lower()
    if l in ("true", "t", "1"):
        return True
    if l in ("false", "f", "0"):
        return False
    raise Exception(
        "Unable to convert string '%s'"
        "to a boolean value" % s
    )
```



One of the hardest problems in Computer Science

```
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
    )
```


Stanford University



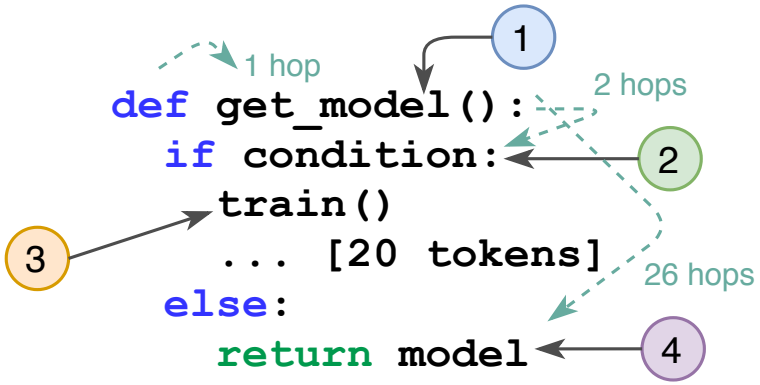
Machine Learning for Code

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

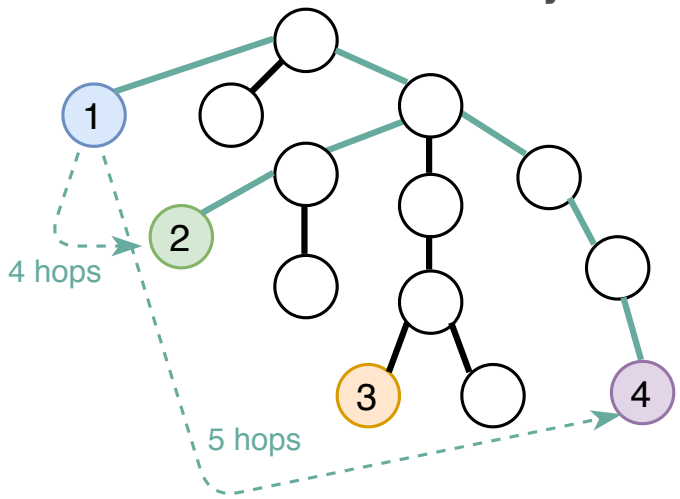
Structure and Context Representation

Source Code as Sequence of Tokens



Context

Source Code as Abstract Syntax Tree



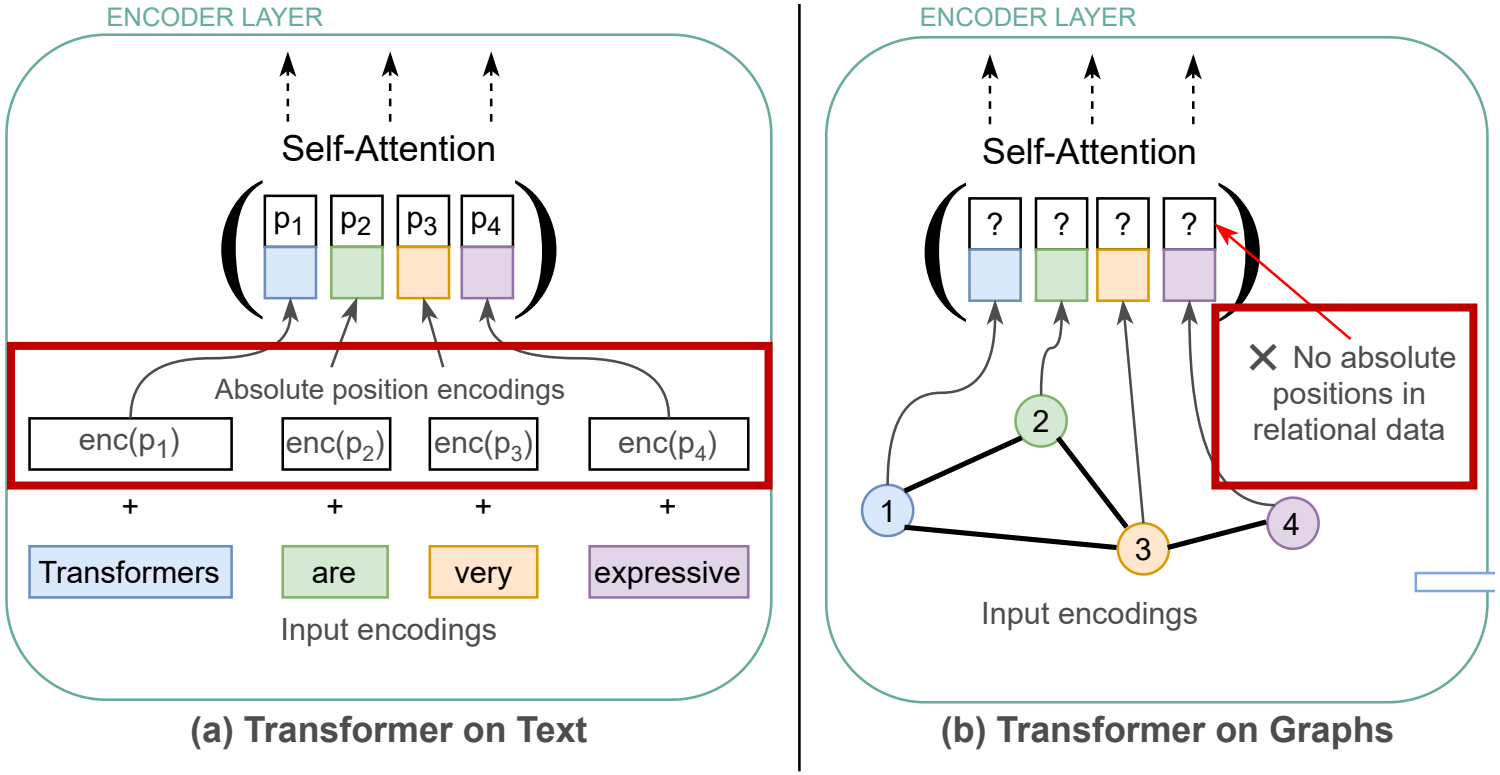
Structure

Structure and **Context** are **complementary representations** of a program.

Most works **leverage only one** of them. We propose to **combine them**.

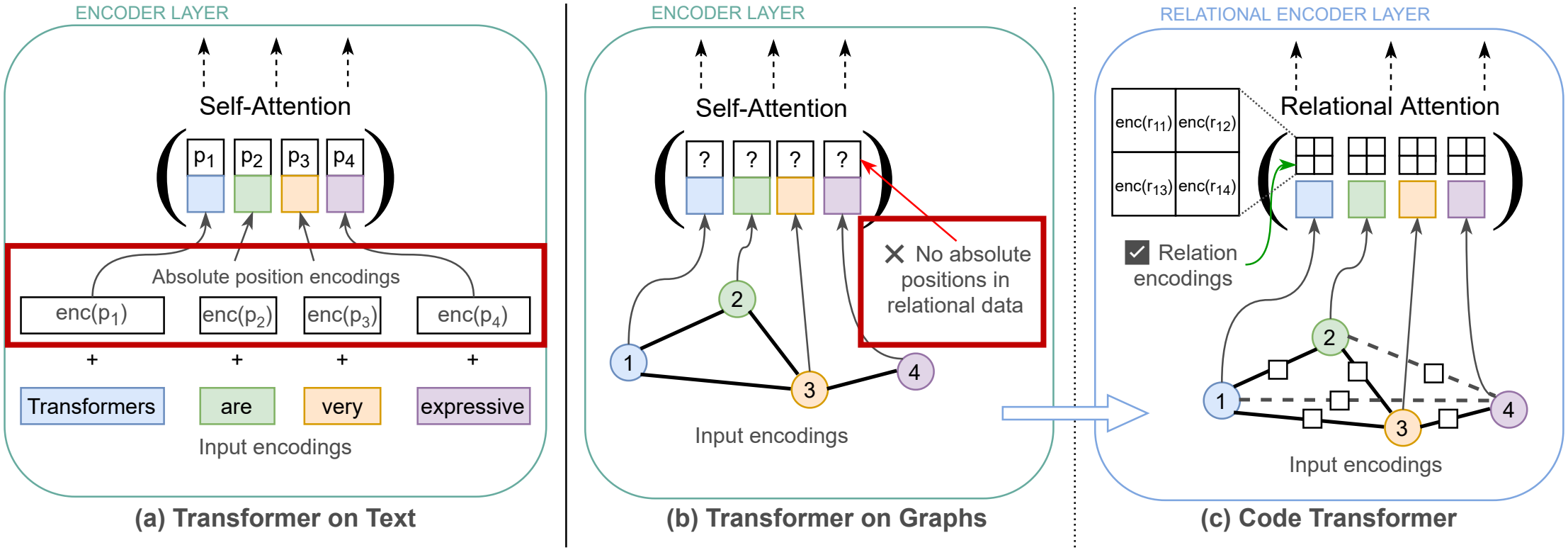
Stanford University

Transformers on Structured Data



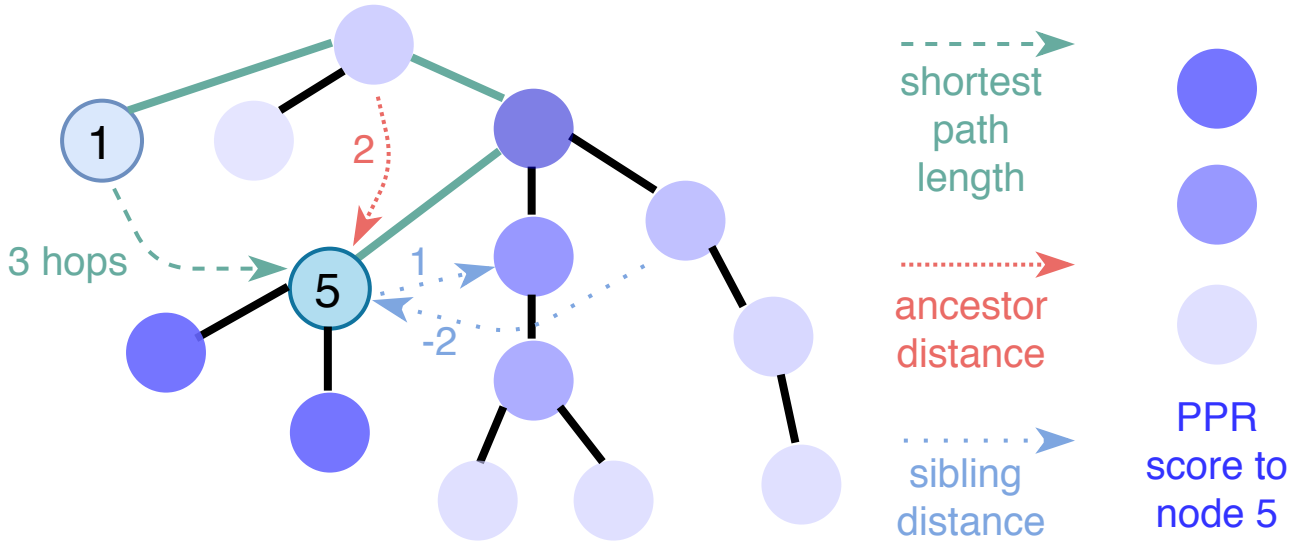
Stanford University

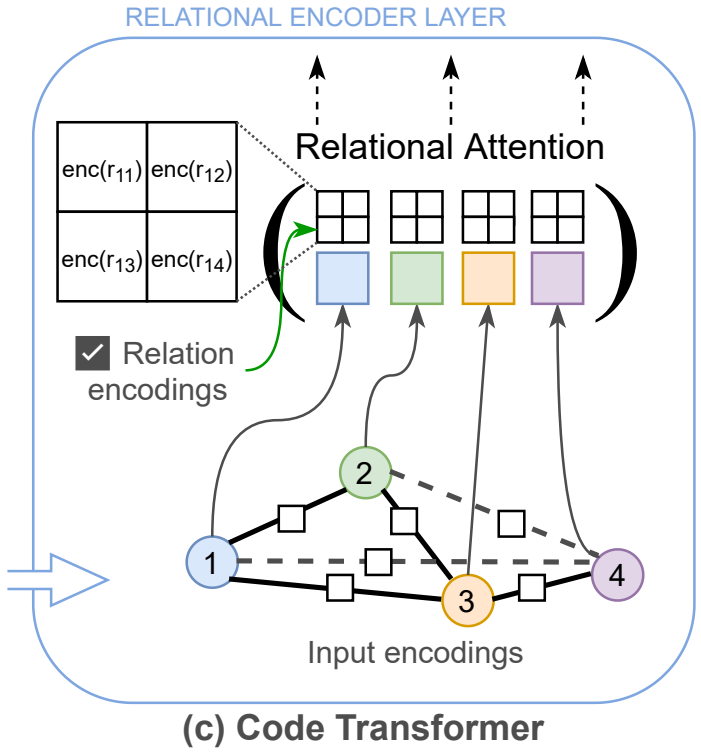
Transformers on Structured Data



Stanford University

Relative Distances on the AST





(c) Code Transformer

Instead of **proprietary, programming-language-specific** pre-processing, we only use **language-agnostic** features from the AST.

Stanford University

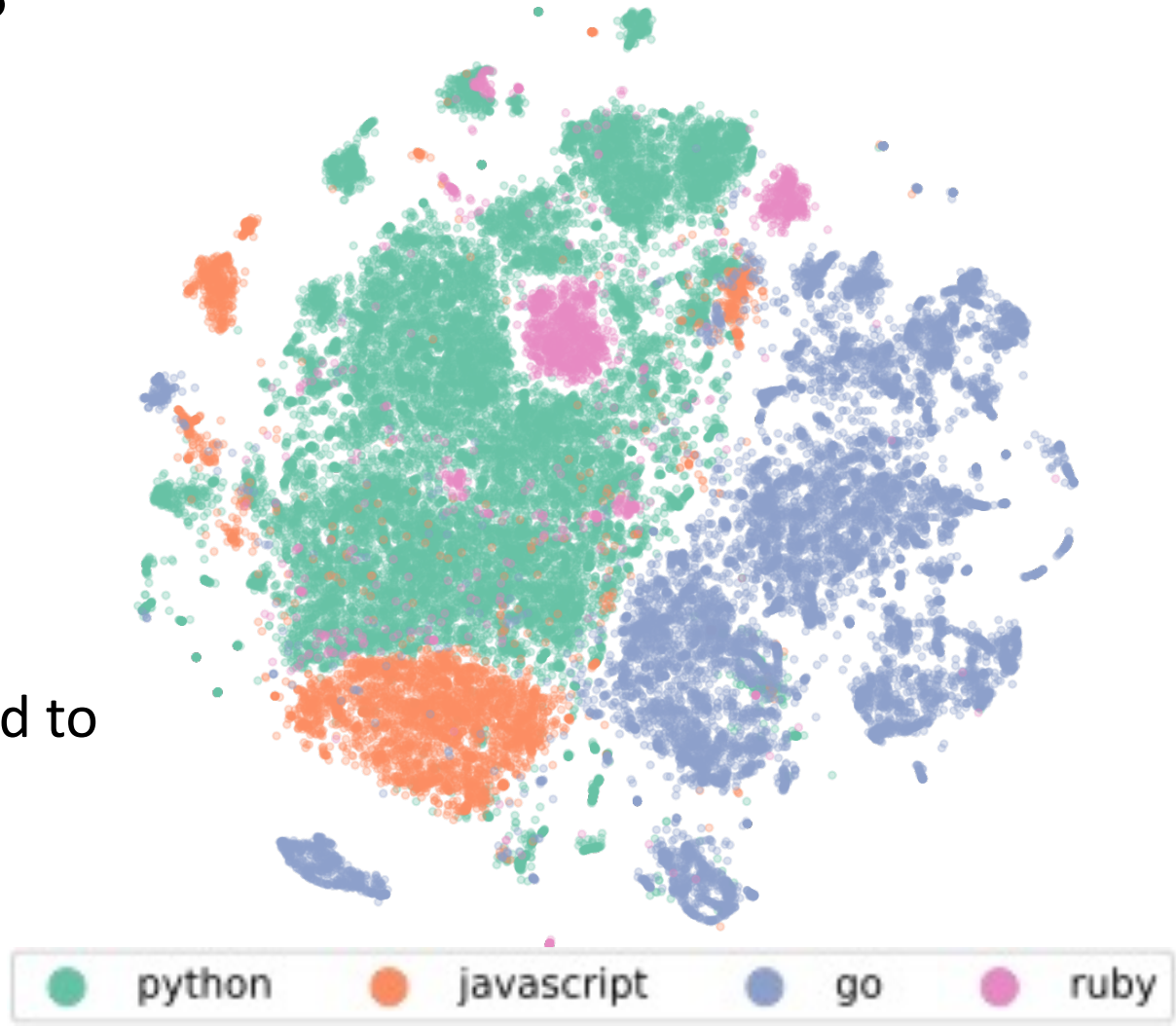
Results: Code Summarization

Model	Python			Javascript			Ruby			Go		
	Prec.	Rec.	F1	Prec.	Rec.	F1	Prec.	Rec.	F1	Prec.	Rec.	F1
code2seq	35.79	24.85	29.34	30.18	19.88	23.97	23.23	10.31	14.28	52.30	43.43	47.45
GREAT	35.07	31.59	33.24	31.20	26.84	28.86	24.64	22.23	23.38	50.01	46.51	48.20
Code Transformer	**36.40**	**33.66**	**34.97**	**35.06**	**29.61**	**32.11**	**31.42**	**24.46**	**27.50**	**55.51**	**48.05**	**51.34**
Code Transformer (Multilanguage)	**38.89**	**33.82**	**36.18**	**36.95**	**29.98**	**33.19**	**33.93**	**28.94**	**31.24**	**56.00**	**50.44**	**53.97**
Code Transformer (Mult. + Pretrain)	**39.67**	**35.29**	**37.35**	**37.06**	**31.94**	**34.31**	**35.19**	**29.36**	**32.01**	**57.73**	**51.89**	**54.65**

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**.



Stanford University

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

Stanford University