[1] DAML Group, Technical University of Munich
[2] SNAP, Stanford University

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]

tl;dr

- We set the **new SOTA on code summarization** by learning from **Structure** and **Context** of **programs**.
- Our **language-agnostic design** enables joint training on **multiple programming languages**.
- Our novel **multi-language training** substantially **improves performance** on each individual language.

Task: Code Summarization

- Predict a method's name given its body.
- A common task to evaluate ML models' performance on code.
- In high-quality code bases, a method's name summarizes its functionality.
- Labels come 'for free', and there is lots of high-quality open-source code.

Example

```
def [MASK](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
    )
```

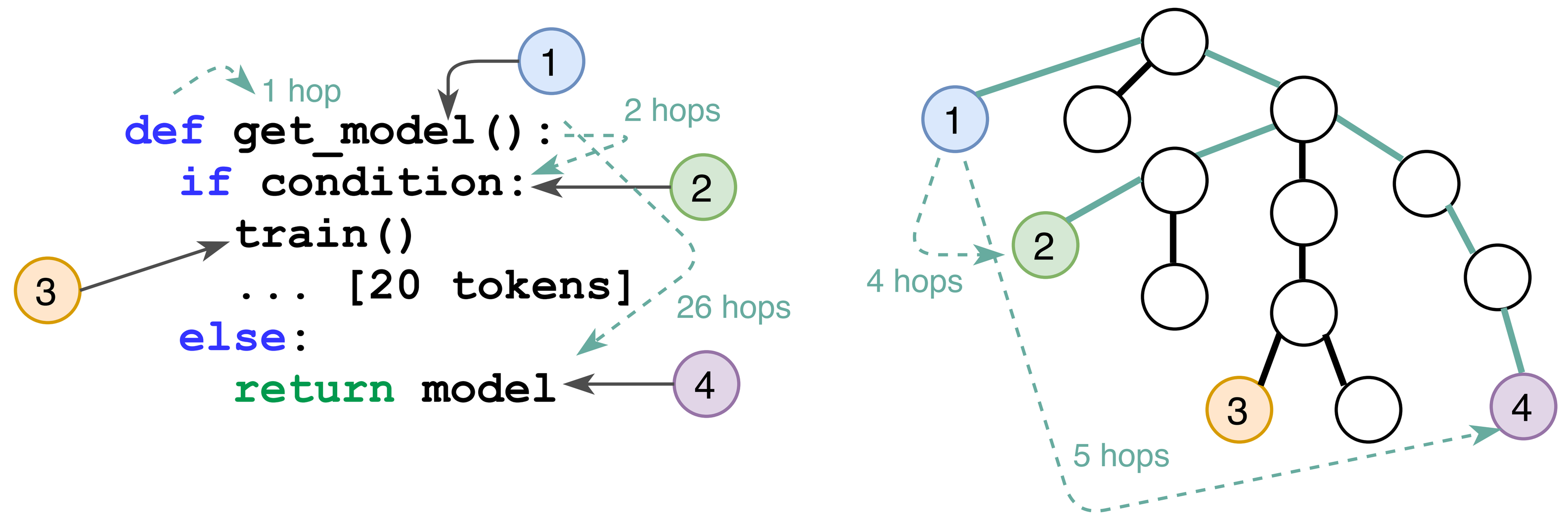
Predictions:

```
(0.8)  parse_bool
(0.1)  to_lower
...
(0.01) get_string
```

Structure and Context Representation

Structure and Context are complementary representations of a program.

	Source code as sequence of tokens (referred to as Context)	Source code as Abstract Syntax Tree (AST; referred to as Structure)
Typical models:	LSTMs, Transformers	GNNs, LSTMs on AST paths

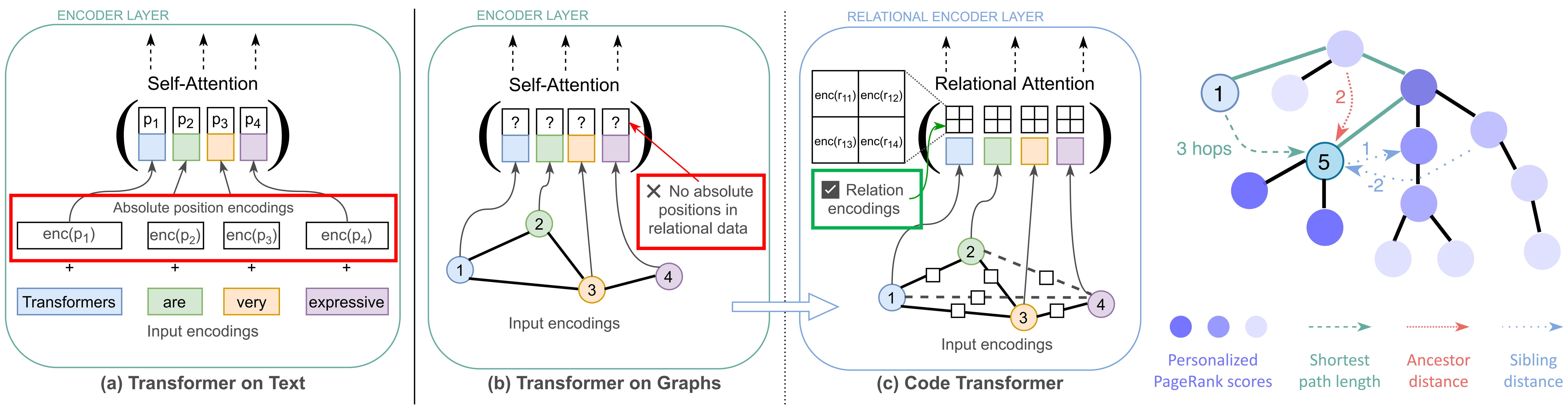


- For instance, `get_model()` is far away from the return statement in the Context; it is hard for a model to capture such long-range dependencies.
- In the AST, the two corresponding nodes are only a few hops apart.
- Many previous works only learn from either Structure or Context.
- We propose the Code Transformer, which learns jointly from both.

Relative Distances on the AST

- Can be computed from the AST for any programming language.
- Capture the local and global structure in the AST.
- Language-agnostic design instead of specialized proprietary pipelines.
- Enables training on any programming language, even jointly (see results).
- Efficient binning enables using continuous-valued distances (e.g., PPR).

Code Transformer



(a) Transformer on Text (b) Transformer on Graphs (c) Code Transformer

Results

Table: Code Summarization Results on the CodeSearchNet dataset (F1 score).

	Model	Python	Javascript	Ruby	Go
Single-language	code2seq	29.3	24.0	14.3	47.5
	GREAT	33.2	28.9	23.4	48.2
	Code Transformer	**35.0**	**32.1**	**27.5**	**51.3**
Multi-language	code2seq*	29.3	26.1	19.9	48.2
	Change (Δ)	-0.02	+2.09	+5.65	+0.72
	GREAT*	33.9	30.4	26.9	50.4
	Change (Δ)	+0.70	+1.55	+3.51	+2.18
	Code Transformer	**36.2**	**33.2**	**31.2**	**53.0**
	Change (Δ)	+1.21	+1.08	+3.74	+2.63
Mult. + Pretrain	Code Transformer	**37.4**	**34.3**	**32.0**	**54.7**
	Change (Δ)	+2.38	+2.20	+4.51	+3.31

* Adapted by us for multilanguage training

- Our Code Transformer outperforms all baselines in all scenarios.
- Multilanguage training improves results on each individual language.
- This is true for all studied models using the Structure, opening exciting directions for future research. No improvement for Context-only models.
- Strongest gains are on Ruby, the language with fewest training samples.

Ablation study

Table: Ablation of Structure and Context (Java-small dataset)

AST distance	F1 score
Code Transformer + Pretrain	**53.77**
Code Transformer	**52.22**
− Structure	50.34
− Context	49.45
− Pointer Network	48.50

Table: Ablation of AST distances w/o Context (Java-small dataset)

AST distance	F1 score
Sibling shortest paths	46.17
Ancestor shortest paths	47.89
Shortest paths	47.76
Personalized PageRank	48.47
All of the above	**48.75**

Multilanguage embeddings

- We obtain a shared embedding space for multiple programming languages.
- Similar methods tend to be mapped to regions close-by in embedding space.
- We can 'measure' similarity of programming languages via their embeddings.

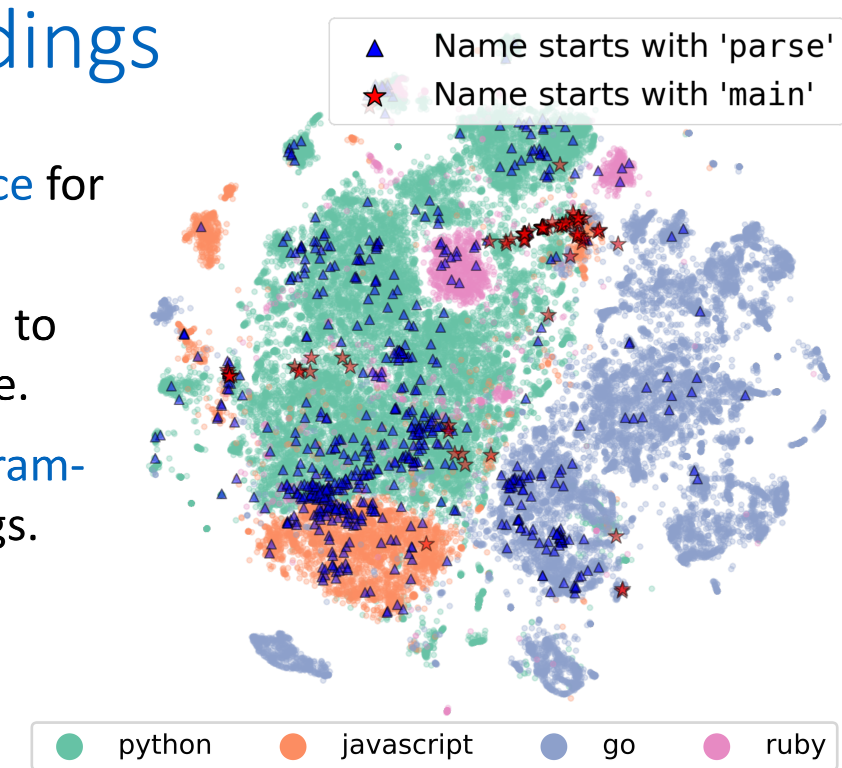


Figure: t-SNE visualization of multilanguage embeddings