GraphTyper: Neural Types Inference from Code Represented as Graph

German Arutyunov
gaarutyunov@edu.hse.ru
https://github.com/gaarutyunov/

Sergey Avdoshin
savdoshin@hse.ru
https://www.hse.ru/staff/avdoshin/

HSE University, 20, Myasnitskaya st., Moscow, Russia

May 28, 2024

Introduction

Task definition, practical relevance and results

- Task: Predicting type annotations in dynamically-typed languages (Python)
- Relevance:
  - Errors due to wrong or absent type annotations
  - Model flexibility for future work
- Results:
  - Masked transformer model that can be directly used on code represented as graph

Introduction

Structure

1. Previous Work
2. Proposed Solution
3. Experiment Results and Ablation Analysis
4. Final Model Quantitative Results
5. Limitations and Workarounds

Previous Work

Main works

Authors	Year	Title	Adoptions
Allamanis et al.	2020	Typilus: Neural Type Hints	Dataset, Metrics
Kim et al.	2022	Pure Transformers are Powerful Graph Learners	Model Architecture

Previous Work

Works for result comparison

Authors	Year	Title	Model Name
Allamanis et al.	2020	Typilus: Neural Type Hints	Typilus (GNN)
Pradel et al.	2020	TypeWriter: neural type prediction with search-based validation	TypeWriter (RNN)
Mir et al.	2021	Type4py: Deep similarity learning-based type inference for python	Type4py (RNN)
Jesse et al.	2021	Learning type annotation: is big data enough?	TypeBERT (Transformer)
Peng et al.	2023	Generative Type Inference for Python	TypeGen (Transformer)

Proposed Solution

Dataset

- 118,440 files with 5,997,459 symbols
- Top 10 types cover half of the dataset
- Only 158 types have over 100 annotations
- The majority are used fewer than 100 times each, forming 32% of the dataset

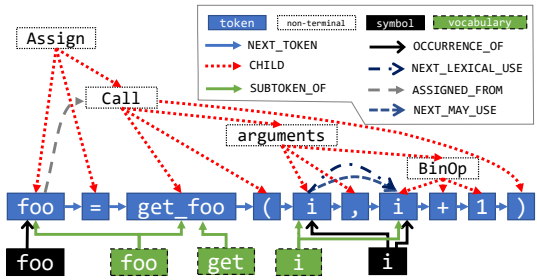
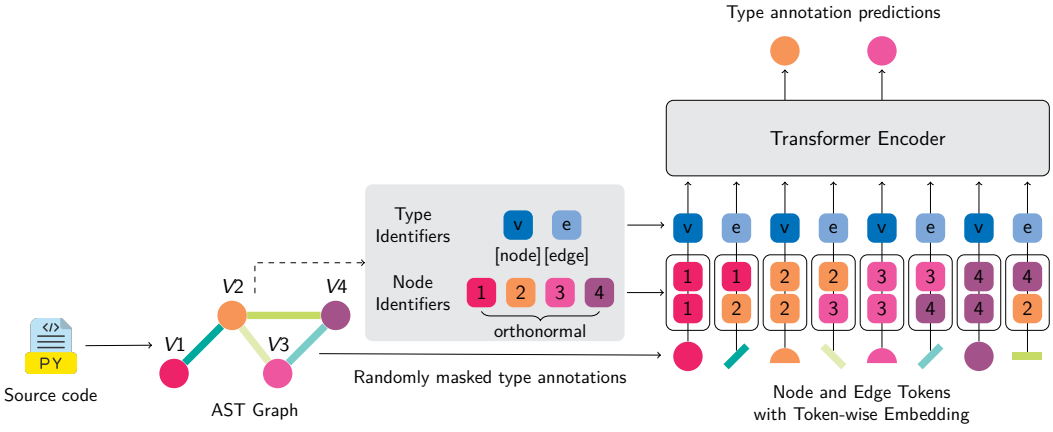


Figure: Sample graph for foo=get_foo(i, i+1) showing different node categories and edge labels.

Proposed Solution

Model architecture



Proposed Solution

Metrics

To test the model, we use two metrics from the Typilus paper:

- Exact Match: Predicted and ground truth types match exactly.
- Match up to Parametric Type: Exact match when ignoring all type parameters.

Experiment Results and Ablation Analysis

Hypothesis

1. Validating the necessity of node and type identifiers that encode graph structure
2. Using the model without node type annotations
3. Increasing the number of parameters
4. Testing different context length
5. Testing different Transformer architectures

Experiment Results and Ablation Analysis

Results Discussion

Top-n		Top-1		Top-3		Top-5
Metric	EM	UTPT	EM	UTPT	EM	UTPT
Plane Transformer	10.15	19.46	15.06	29.40	16.81	37.91
+ Node & Type Identifiers	30.88	36.55	40.33	50.37	42.82	56.01
+ Type Annotations	33.36	**42.28**	41.71	52.90	43.62	57.00
+ Decoder (Autoencoder)	15.90	16.65	28.26	32.81	44.17	56.33
or Longer Context	**38.49**	39.80	**53.14**	**57.41**	**58.80**	**67.38**
or More Parameters	29.39	31.82	44.85	49.72	49.74	56.14

Final Model Quantitative Results

Comparison with previous work

Top-n		Top-1		Top-3		Top-5
Metric	EM	UTPT	EM	UTPT	EM	UTPT
GraphTyper	34.71	36.43	45.47	55.02	50.70	64.58
TypeBERT	45.40	48.10	51.40	53.50	54.10	56.50
TypeWriter	56.10	58.30	63.70	67.30	65.90	70.40
Typilus	66.10	74.20	71.60	79.80	72.70	80.90
Type4Py	75.80	80.60	78.10	83.80	78.70	84.70
TypeGen	**79.20**	**87.30**	**85.60**	**91.00**	**87.00**	**91.70**

Limitations and Workarounds

Considerations for future work

- Type Vocabulary Size
- Absence of Natural Language Information

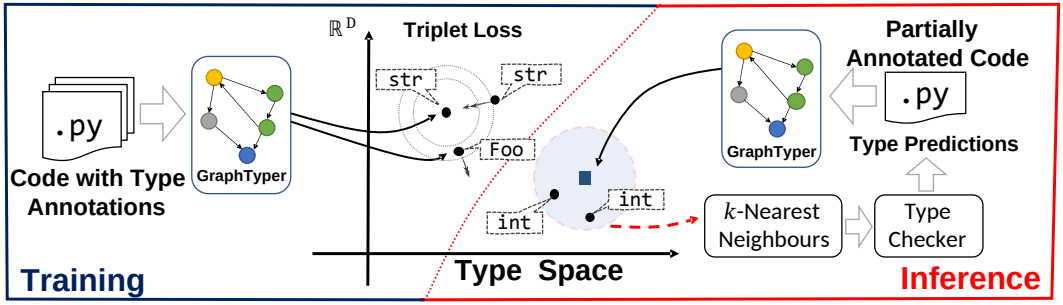


Figure: Solution to the problem using Deep Similarity Learning.

Conclusion

- Main accomplishment:
  - A universal Transformer model that can be applied on code represented as graph
- Future Work:
  - Universal code graph representation
  - Detecting duplicates
  - Code and docstring generation
  - Vulnerability and error detection
  - Refactoring

Acknolegements

This research was supported in part through computational resources of HPC facilities at HSE University [1]

[1] P. S. Kostenetskiy, R. A. Chulkevich, and V. I. Kozyrev. "HPC Resources of the Higher School of Economics." In: Journal of Physics: Conference Series 1740.1 (Jan. 2021)