Neural Metrics

Yegor Bugayenko

Lecture #24 out of 24
80 minutes

Michael Pradel and Satish Chandra

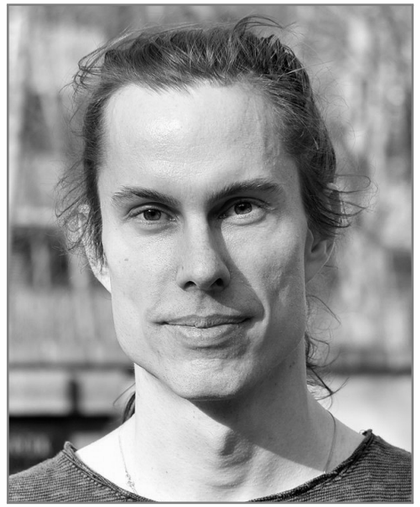
"Neural software analysis offers a fresh approach, enhancing or even surpassing traditional program analysis in some areas"

— Michael Pradel and Satish Chandra. Neural software analysis. *Communications of the ACM*, 2022. doi:10.1145/3460348

Pavol Bielik

"In this paper we present a new, automated approach for creating static analyzers: instead of manually providing the various inference rules of the analyzer, the key idea is to learn these rules from a dataset of programs."

— Pavol Bielik, Veselin Raychev, and Martin Vechev. Learning a Static Analyzer from Data. In *Proceedings of the 29th International Conference on Computer Aided Verification (CAV)*, pages 233–253. Springer, 2017. doi:10.1007/978-3-319-63387-9_12



Tomas Mikolov

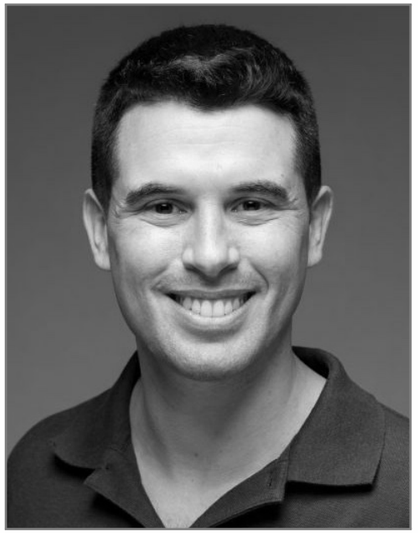
“The meanings of ‘Canada’ and ‘Air’ cannot be easily combined to obtain ‘Air Canada’. Motivated by this example, we present a simple method for finding phrases in text, and show that learning good vector representations for millions of phrases is possible.”

—

Koushik Sen

“This paper presents DeepBugs, a learning approach to name-based bug detection, which reasons about names based on a semantic representation and which automatically learns bug detectors instead of manually writing them. We formulate bug detection as a binary classification problem and train a classifier that distinguishes correct from incorrect code.”

—



Uri Alon

"**code2vec**: The main idea is to represent a code snippet as a single fixed-length code vector, which can be used to predict semantic properties of the snippet."

— Uri Alon, Meital Zilberstein, Omer Levy, and Eran Yahav. Code2vec: Learning Distributed Representations of Code. *Proceedings of the Principles of Programming Languages (POPL)*, 2019. doi:10.1145/3290353

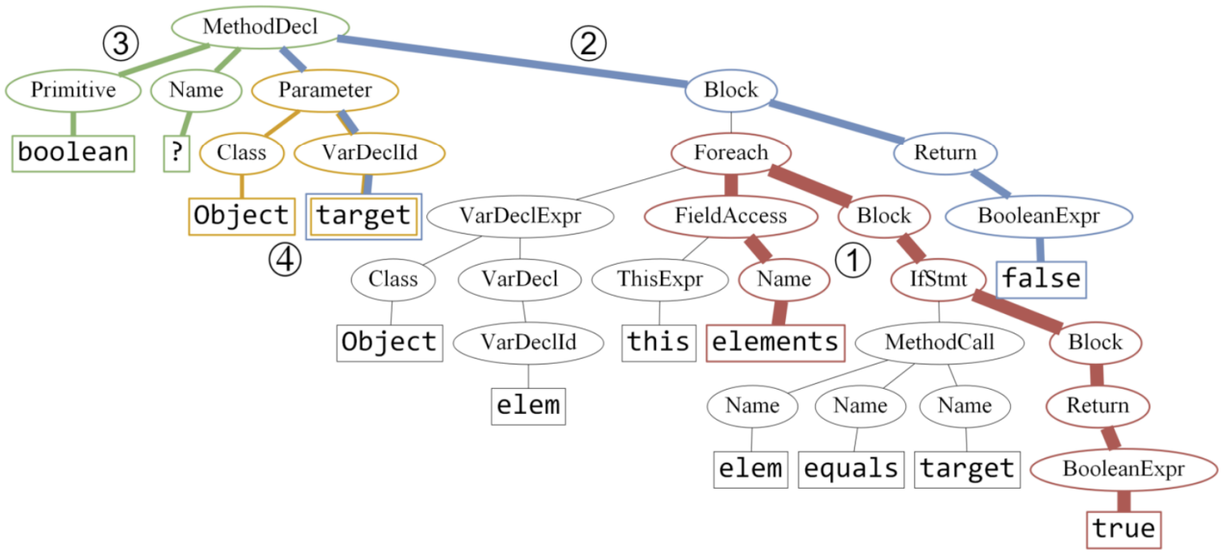


Fig. 3. The top-4 attended paths of Figure 2a, as were learned by the model, shown on the AST of the same snippet. The width of each colored path is proportional to the attention it was given (**red** ①: **0.23**, **blue** ②: **0.14**, **green** ③: **0.09**, **orange** ④: **0.07**).

Source: Uri Alon, Meital Zilberstein, Omer Levy, and Eran Yahav. Code2vec: Learning Distributed Representations of Code. *Proceedings of the Principles of Programming Languages (POPL)*, 2019. doi:10.1145/3290353

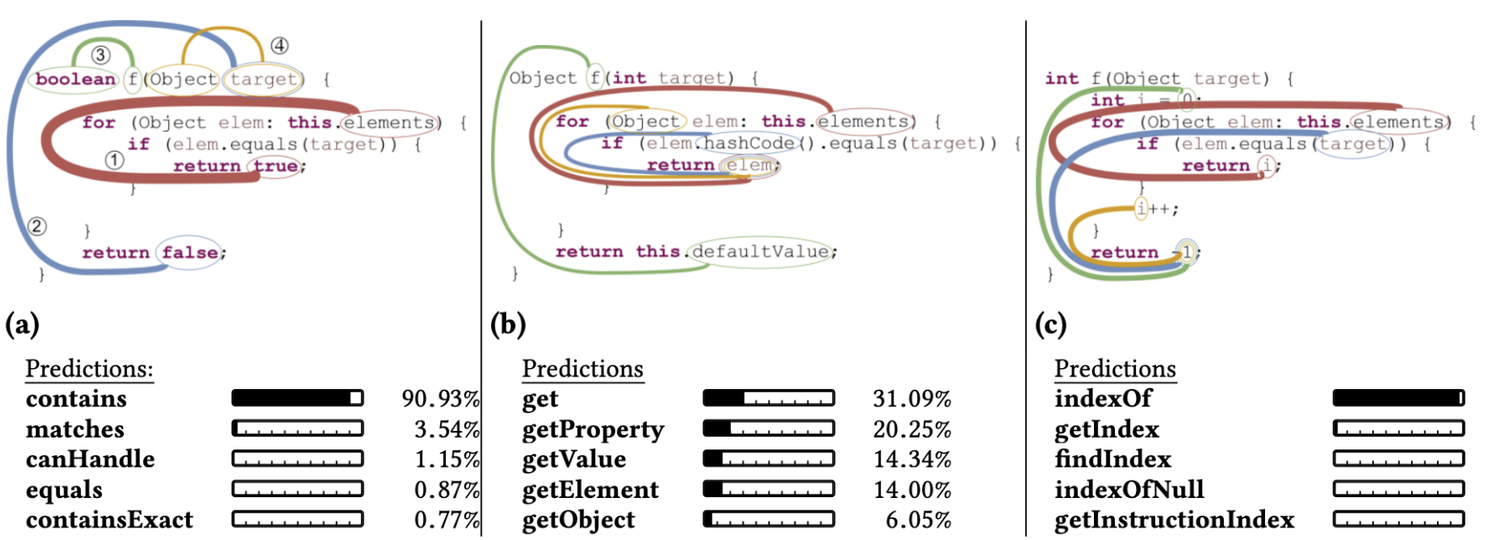


Fig. 2. Examples of three methods that can be easily distinguished by our model despite having similar syntactic structure: our model successfully captures the subtle differences between them and predicts meaningful names. Each method portrays the top-4 paths that were given the most attention by the model. The widths of the colored paths are proportional to the attention that each path was given.

Source: Uri Alon, Meital Zilberstein, Omer Levy, and Eran Yahav. Code2vec: Learning Distributed Representations of Code. *Proceedings of the Principles of Programming Languages (POPL)*, 2019. doi:10.1145/3290353

Mohammad Mahdi Mohajer

"SkipAnalyzer consists of three components, 1) an LLM-based static bug detector that scans source code and reports specific types of bugs, 2) an LLM-based false positive filter, and 3) an LLM-based patch generator that can generate patches for the detected bugs above. As a proof-of-concept, SkipAnalyzer is built on ChatGPT."

— Mohammad Mahdi Mohajer, Reem Aleithan, Nima Shiri Harzevili, Moshi Wei, Alvine Boaye Belle, Hung Viet Pham, and Song Wang. SkipAnalyzer: An Embodied Agent for Code Analysis With Large Language Models. *ArXiv*, 2023. doi:10.48550/arXiv.2310.18532

Haonan Li

"UBITect produces many false positives from the static analysis. With a pilot study of 20 false positives, we can successfully prune 8 out of 20 based on GPT-3.5, whereas GPT-4 had a near-perfect result of 16 out of 20."

— Haonan Li, Yu Hao, Yizhuo Zhai, and Zhiyun Qian. Assisting Static Analysis With Large Language Models: A ChatGPT Experiment. In *Proceedings of the 31st Joint European Software Engineering Conference and Symposium on the Foundations of Software Engineering*, pages 2107–2111, 2023. doi:10.1145/3611643.3613078

Elif Nur Haner Kirğil

“In the study, the cohesion value, which is one of the most important criteria for evaluating software quality, was predicted by RF, KNN, REPTree, SVM, MLP, and LR machine learning techniques.”

—

Table of Contents

No

No, they cannot.

LLaMa 65B (4-bit GPTQ) model: 1 false alarms in 15 good examples. Detects 0 of 13 bugs.
Baize 30B (8-bit) model: 0 false alarms in 15 good examples. Detects 1 of 13 bugs.
Galpaca 30B (8-bit) model: 0 false alarms in 15 good examples. Detects 1 of 13 bugs.
Koala 13B (8-bit) model: 0 false alarms in 15 good examples. Detects 0 of 13 bugs.
Vicuna 13B (8-bit) model: 2 false alarms in 15 good examples. Detects 1 of 13 bugs.
Vicuna 7B (FP16) model: 1 false alarms in 15 good examples. Detects 0 of 13 bugs.

GPT 3.5: 0 false alarms in 15 good examples. Detects 7 of 13 bugs.
GPT 4: 0 false alarms in 15 good examples. Detects 13 of 13 bugs.

Source: Chris Taylor. Can Open-Source LLMs Detect Bugs in C++ Code? https://catid.io/posts/llm_bugs/, 2023. [Online; accessed 15-03-2024]

References

Uri Alon, Meital Zilberstein, Omer Levy, and Eran Yahav. Code2vec: Learning Distributed Representations of Code. *Proceedings of the Principles of Programming Languages (POPL)*, 2019. doi:10.1145/3290353.

Pavol Bielik, Veselin Raychev, and Martin Vechev. Learning a Static Analyzer from Data. In *Proceedings of the 29th International Conference on Computer Aided Verification (CAV)*, pages 233–253. Springer, 2017. doi:10.1007/978-3-319-63387-9_12.

Haonan Li, Yu Hao, Yizhuo Zhai, and Zhiyun Qian. Assisting Static Analysis With Large Language Models: A ChatGPT Experiment. In *Proceedings of the 31st Joint European Software Engineering Conference and Symposium on the Foundations of Software Engineering*, pages 2107–2111, 2023. doi:10.1145/3611643.3613078.

Mohammad Mahdi Mohajer, Reem Aleithan, Nima Shiri Harzevili, Moshi Wei, Alvine Boaye Belle, Hung Viet Pham, and Song Wang. SkipAnalyzer: An Embodied Agent for Code Analysis With Large Language Models. *ArXiv*, 2023. doi:10.48550/arXiv.2310.18532.

Michael Pradel and Satish Chandra. Neural software analysis. *Communications of the ACM*, 2022. doi:10.1145/3460348.

Chris Taylor. Can Open-Source LLMs Detect Bugs in C++ Code? `https://catid.io/posts/llm_bugs/`, 2023. [Online; accessed 15-03-2024].