Neural Metrics

Yegor Bugayenko

Lecture #24 out of 24
80 minutes

The slidedeck was presented by the author in this YouTube Video

1. Non-deterministic methods of code analysis may be more effective than symbolic execution or data flow analysis.

Michael Pradel

"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*, 65(1):86–96, 2021. doi:10.1145/3460348

> 2. Machine Learning is the most obvious non-deterministic method of code analysis.

Francesca Arcelli Fontana

"We conclude that the application of machine learning to the detection of [these] code smells can provide high accuracy (>96%), and only a hundred training examples are needed to reach at least 95% accuracy."

— Francesca Arcelli Fontana, Mika V. Mäntylä, Marco Zanoni, and Alessandro Marino. Comparing and Experimenting Machine Learning Techniques for Code Smell Detection. *Empirical Software Engineering*, 21(1):1143–1191, 2016. doi:10.1007/s10664-015-9378-4

Martin White

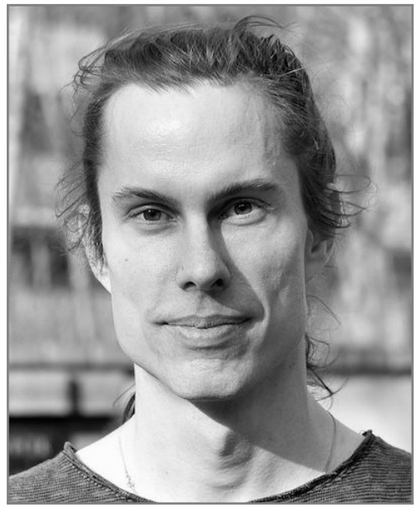
"Among the true positives, we found pairs mapping to all four clone types. We compared our approach to a traditional structure-oriented technique and found that our learning-based approach detected clones that were either undetected or suboptimally reported by the prominent tool Deckard."

— Martin White, Michele Tufano, Christopher Vendome, and Denys Poshyvanyk. Deep Learning Code Fragments for Code Clone Detection. In *Proceedings of the 31st International Conference on Automated Software Engineering*, pages 87–98, 2016. doi:10.1145/2970276.2970326

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."

— Tomas Mikolov, Ilya Sutskever, Kai Chen, Greg S. Corrado, and Jeff Dean. Distributed Representations of Words and Phrases and Their Compositionality, 2013

Koushik Sen

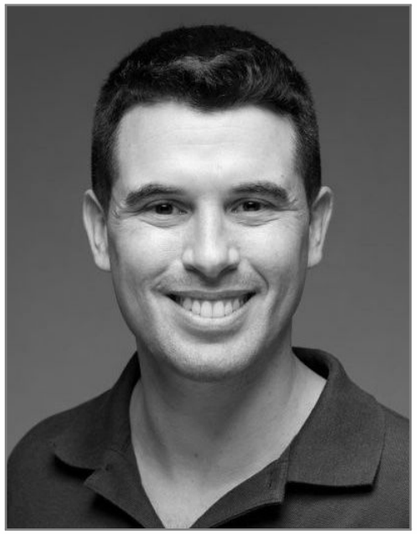
"This paper presents DeepBugs, a learning approach to name-based bug detection, which reasons about names based on a semantic representation and which automatically learns bug detectors instead of manually writing them. We formulate bug detection as a binary classification problem and train a classifier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

— Michael Pradel and Koushik Sen. DeepBugs: A Learning Approach to Name-Based Bug Detection. *Proceedings of the ACM on Programming Languages*, 2(1):1–25, 2018. doi:10.1145/3276517

Dario Di Nucci

"While Arcelli Fontana et al. [2016] opened a new perspective for code smell detection, in the context of our research we found a number of possible limitations that might threaten the results of this study. Our findings show that the high performance achieved in their study was in fact due to the specific dataset employed rather than the actual capabilities of machine-learning techniques for code smell detection."

— Dario Di Nucci, Fabio Palomba, Damian A. Tamburri, Alexander Serebrenik, and Andrea De Lucia. Detecting Code Smells Using Machine Learning Techniques: Are We There yet? In *Proceedings of the 25th International Conference on Software Analysis, Evolution and Reengineering (SANER)*, pages 612–621. IEEE, 2018. doi:10.1109/SANER.2018.8330266



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)*, 3(1), 2019. doi:10.1145/3290353

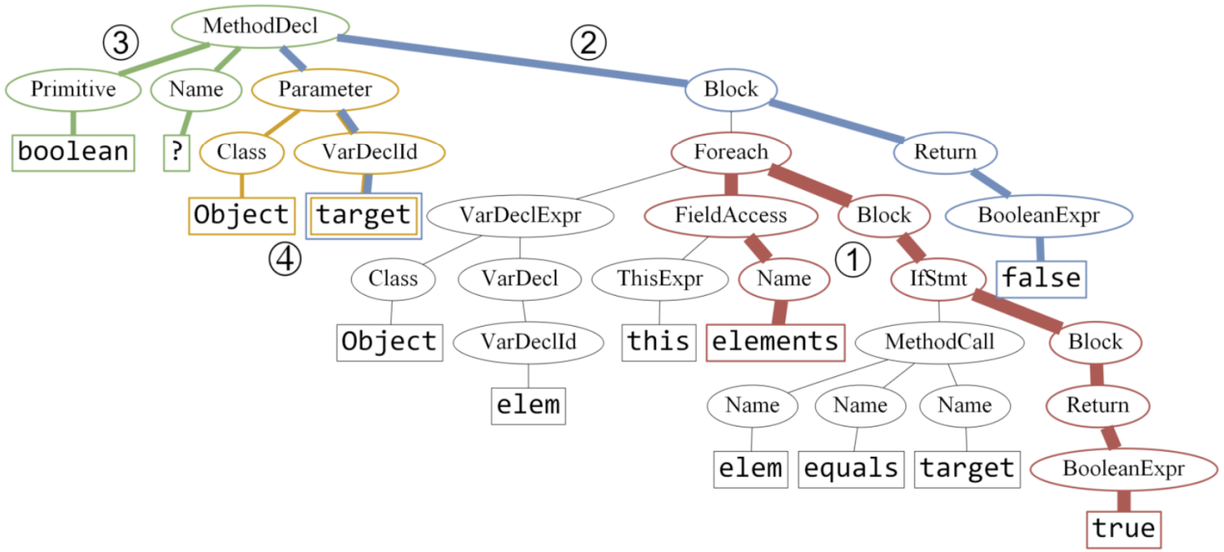


Fig. 3. The top-4 attended paths of Figure 2a, as were learned by the model, shown on the AST of the same snippet. The width of each colored path is proportional to the attention it was given (**red** ①: **0.23**, **blue** ②: **0.14**, **green** ③: **0.09**, **orange** ④: **0.07**).

Source: Uri Alon, Meital Zilberstein, Omer Levy, and Eran Yahav. Code2vec: Learning Distributed Representations of Code. *Proceedings of the Principles of Programming Languages (POPL)*, 3(1), 2019. doi:10.1145/3290353

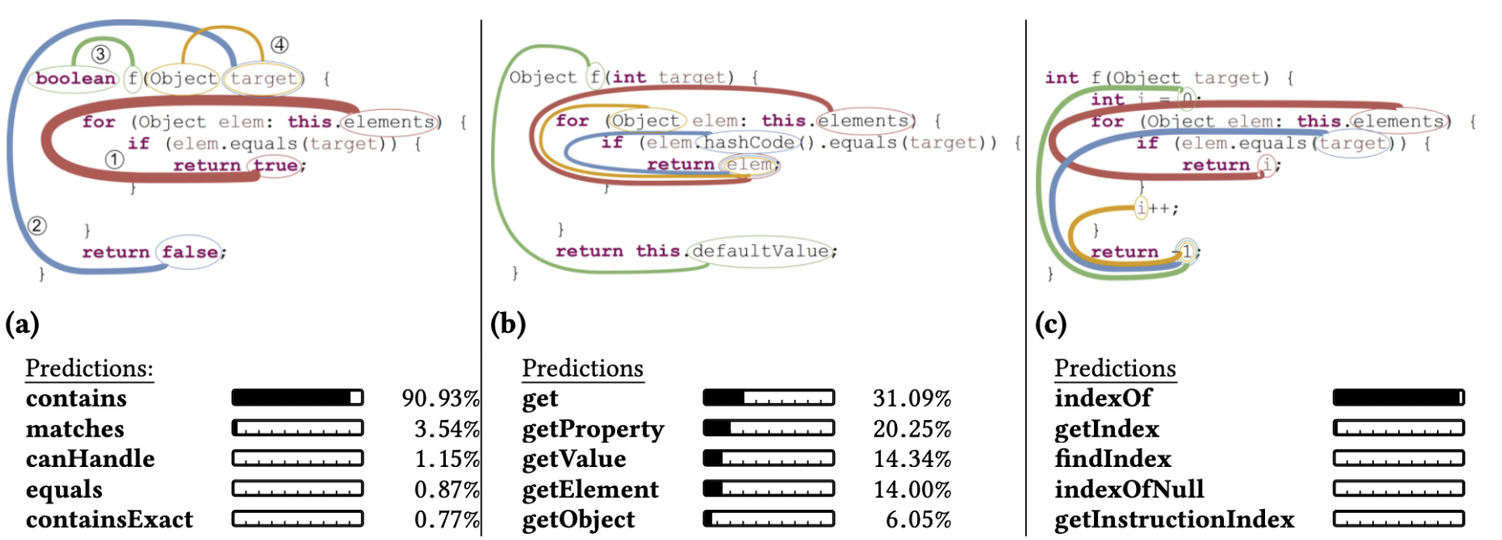


Fig. 2. Examples of three methods that can be easily distinguished by our model despite having similar syntactic structure: our model successfully captures the subtle differences between them and predicts meaningful names. Each method portrays the top-4 paths that were given the most attention by the 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)*, 3(1), 2019. doi:10.1145/3290353

3. Since 2023, Large Language Models (LLM) started to demonstrate code analysis capabilities.

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, 2023

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."

— Elif Nur Haner Kırğıl and Tülin Erçelebi Ayyıldız. Predicting Software Cohesion Metrics With Machine Learning Techniques. *Applied Sciences*, 13(6): 3722, 2023. doi:10.3390/app13063722

Table of Contents

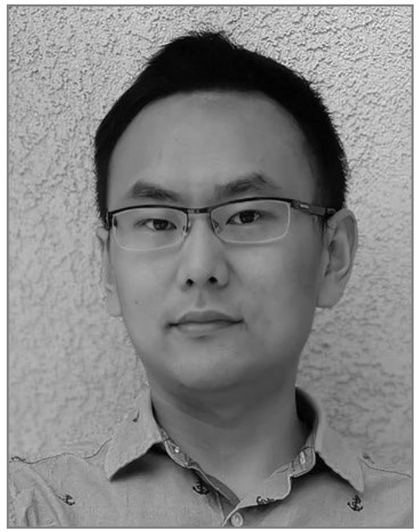
No

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.

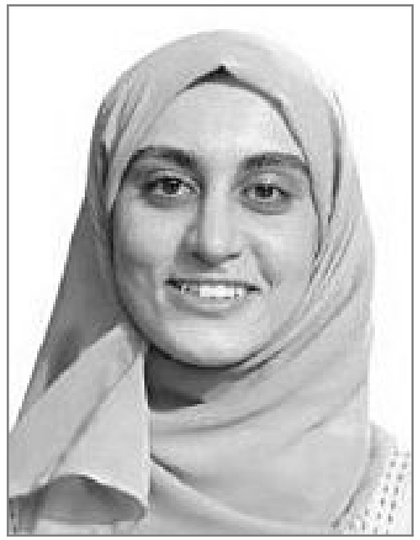
Source: Chris Taylor. Can Open-Source LLMs Detect Bugs in C++ Code? https://catid.io/posts/llm_bugs/, 2023. [Online; accessed 15-03-2024]



Zhiyun Qian

"Our real-world evaluations identified four previously undiscovered UBI (Use Before Initialization) bugs in the mainstream Linux kernel, which the Linux community has acknowledged. This study reaffirms the potential of marrying static program analysis with LLMs, setting a compelling direction for future research in this area."

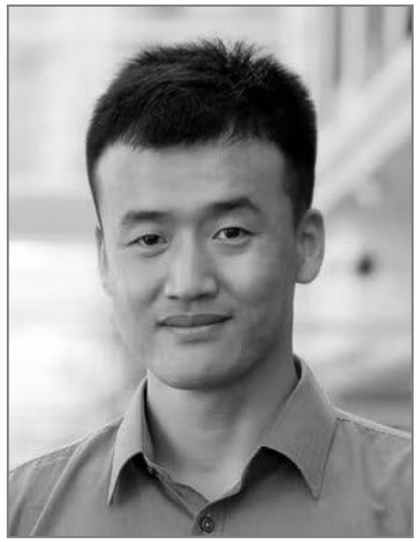
— Haonan Li, Yu Hao, Yizhuo Zhai, and Zhiyun Qian. Enhancing Static Analysis for Practical Bug Detection: An LLM-Integrated Approach. *Proceedings of the ACM on Programming Languages*, 8(1):474–499, 2024. doi:10.1145/3649828



Reem Aleithan

"Our study demonstrates that ChatGPT can achieve remarkable performance in the mentioned static analysis tasks, including bug detection and false-positive warning removal."

— Mohammad Mahdi Mohajer, Reem Aleithan, Nima Shiri Harzevili, Moshi Wei, Alvine Boaye Belle, Hung Viet Pham, and Song Wang. Effectiveness of ChatGPT for Static Analysis: How Far Are We? In *Proceedings of the 1st International Conference on AI-Powered Software*, pages 151–160, 2024. doi:10.1145/3711816



Chengnian Sun

"LATTE combines code snippets to construct prompts. LATTE detected a total of 119 unique bugs, including 37 previously unknown bugs (10 CVE numbers have been given due to high threat), which outperforms the state of the art."

— Puzhuo Liu, Chengnian Sun, Yaowen Zheng, Xuan Feng, Chuan Qin, Yuncheng Wang, Zhenyang Xu, Zhi Li, Peng Di, Yu Jiang, et al. LLM-Powered Static Binary Taint Analysis. *ACM Transactions on Software Engineering and Methodology*, 34(3), 2025. doi:10.1145/3711816

Bibliography

Uri Alon, Meital Zilberstein, Omer Levy, and Eran Yahav. Code2vec: Learning Distributed Representations of Code. *Proceedings of the Principles of Programming Languages (POPL)*, 3(1), 2019. doi:10.1145/3290353.

Francesca Arcelli Fontana, Mika V. Mäntylä, Marco Zanoni, and Alessandro Marino. Comparing and Experimenting Machine Learning Techniques for Code Smell Detection. *Empirical Software Engineering*, 21(1): 1143–1191, 2016. doi:10.1007/s10664-015-9378-4.

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.

Dario Di Nucci, Fabio Palomba, Damian A. Tamburri, Alexander Serebrenik, and Andrea De Lucia. Detecting Code Smells Using Machine Learning Techniques: Are We There yet? In *Proceedings of the 25th International Conference on Software Analysis, Evolution and Reengineering (SANER)*, pages 612–621. IEEE, 2018. doi:10.1109/SANER.2018.8330266.

Elif Nur Haner Kırğıl and Tülin Erçelebi Ayyıldız. Predicting Software Cohesion Metrics With Machine Learning Techniques. *Applied Sciences*, 13(6):3722, 2023. doi:10.3390/app13063722.

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.

Haonan Li, Yu Hao, Yizhuo Zhai, and Zhiyun Qian. Enhancing Static Analysis for Practical Bug Detection: An LLM-Integrated Approach. *Proceedings of the ACM on Programming Languages*, 8(1):474–499, 2024. doi:10.1145/3649828.

Puzhuo Liu, Chengnian Sun, Yaowen Zheng, Xuan Feng, Chuan Qin, Yuncheng Wang, Zhenyang Xu, Zhi Li, Peng Di, Yu Jiang, et al. LLM-Powered Static Binary Taint Analysis. *ACM Transactions on Software Engineering and Methodology*, 34(3), 2025. doi:10.1145/3711816.

Tomas Mikolov, Ilya Sutskever, Kai Chen, Greg S. Corrado, and Jeff Dean. Distributed Representations of Words and Phrases and Their Compositionality, 2013.

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, 2023.

Mohammad Mahdi Mohajer, Reem Aleithan, Nima Shiri Harzevili, Moshi Wei, Alvine Boaye Belle, Hung Viet Pham, and Song Wang. Effectiveness of ChatGPT for Static Analysis: How Far Are We? In *Proceedings of the 1st International Conference on AI-Powered Software*, pages 151–160, 2024. doi:10.1145/3711816.

Michael Pradel and Satish Chandra. Neural Software Analysis. *Communications of the ACM*, 65(1):86–96, 2021. doi:10.1145/3460348.

Michael Pradel and Koushik Sen. DeepBugs: A Learning Approach to Name-Based Bug Detection. *Proceedings of the ACM on Programming Languages*, 2(1):1–25, 2018. doi:10.1145/3276517.

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

Martin White, Michele Tufano, Christopher Vendome, and Denys Poshyvanyk. Deep Learning Code Fragments for Code Clone Detection. In *Proceedings of the 31st International Conference on Automated Software Engineering*, pages 87–98, 2016. doi:10.1145/2970276.2970326.