ROBUST SOURCE CODE REPRESENTATION LEARNING

Saikat Chakraborty

March 19, 2021

1 Background and Research Objective

Automating different tasks for software engineers/programmers has long been an interesting topic of research. Recent advancement of Machine Learning (ML), especially Deep Learning(DL), leads the way to automate different Software Engineering(SE) tasks in a data-driven fashion. The availability of vast sources of code and related data in open source repositories and forums makes some of the traditional software engineering tasks suitable for solving with DL. Code Summarization [21, 2, 16, 3, 15, 13, 1, 28], Bug Detection [23, 18, 24, 30, 6], Program Repair [8, 5, 20], Code Translation [7, 10, 26], Clone Detection [29, 27, 25], Code completion [17, 14] are some of the tasks that are suitable for solving with Deep Learning. The main challenge in all these problems is properly learning the representation of code. While most of the approaches to solve SE tasks use task-specific representation learning, some work for learning transferable semantic representation of code. For instance, Code2Vec [4] learns representation of code tokens using AST paths. Our past work, CODIT [5], learns the code embedding for code change prediction tasks with a Probabilistic Context-Free Grammar based translation model. We have also worked on a Code summarization model [1] with transformer, and Code Vulnerability Detection Model [6] with graph neural network, where learning the code-representation is paramount. All these models learn code-representations that are non-transferable and often do not capture all the code information. For example, our model [1] learned an embedding that prioritized the identifier names, vastly ignoring the syntax and semantics in our code summarization work.

When a model learn to solve any SE problem, it has to solve two major sub-problems – *i.e.*, learning the semantics of the data, and learning to solve the problem. For instance, when a model learns to detect vulnerability in source code, it has to solve two tasks – `(i)` it learns what different components of the code mean (learn the code semantics), and `(ii)` it learns to reason about the vulnerability patterns in the code. Although these two tasks are often interleaved in the model, these are two different tasks nonetheless.

Recently, research is moving towards decoupling code representation learning and task-specific learning [9, 19, 22]. A widely accepted research direction is to learn general-purpose representation for code that encompasses all the code information in a task agnostic way. Then refine the model based on the downstream task. A significant advantage in doing so is, in this way, the demand for annotated data in downstream tasks becomes very low. CodeBERT [11] is such a model, pre-trained on bi-modal data to captures the semantic interaction between the input modalities (*i.e.,* code and natural languages). However, being a BERT [9] based model, CodeBERT inherits BERT-style models' inadequacy in generative tasks. GraphCodeBERT [12] improves upon CodeBERT by leveraging data flow in the code. Both CodeBERT, and GraphCodeBERT showed very good performance on different tasks including code search, code summarization, etc. In out current work, we are trying to develop a model that supposedly learns beyond the code's syntax. In other words, the model understands what the meaning of the code is. We trained the model in multiple languages (*e.g.*, `Java`, `Python`), and the model works very well in unseen languages (*e.g.*, `Php`, `Go`, etc.). Our model works very well in a wide variety of SE tasks, including summarization, code generation, code translation, clone detection, etc. My primary research focus is to learn program/source code representation *w.r.t.* three different objectives. `(i)` Representation is transferrable across different tasks and different languages `(ii)` It encompasses major information modalities in the source code (syntax, semantics, etc.), and `(iii)` The models are appropriate for different tasks *w.r.t.* source code.

References

[1] Wasi Ahmad, Saikat Chakraborty, Baishakhi Ray, and Kai-Wei Chang. A transformer-based approach for source code summarization. *Proceedings of the 58th Annual Meeting of the Association for Computational Linguistics (ACL)*, pages 4998–5007, July 2020.

[2] Miltiadis Allamanis, Hao Peng, and Charles A. Sutton. A convolutional attention network for extreme summarization of source code. volume 48 of *JMLR Workshop and Conference Proceedings*, pages 2091–2100. JMLR.org, 2016.

[3] Uri Alon, Shaked Brody, Omer Levy, and Eran Yahav. code2seq: Generating sequences from structured representations of code. International Conference on Learning Representations(ICLR), 2019.

[4] Uri Alon, Meital Zilberstein, Omer Levy, and Eran Yahav. code2vec: Learning distributed representations of code. volume 3, page 40. Proceedings of the ACM on Programming Languages(POPL), ACM, 2019.

[5] Saikat Chakraborty, Yangruibo Ding, Miltiadis Allamanis, and Baishakhi Ray. Codit: Code editing with tree-based neural models. *IEEE Transactions on Software Engineering*, pages 1–1, 2020.

[6] Saikat Chakraborty, Rahul Krishna, Yangruibo Ding, and Baishakhi Ray. Deep learning based vulnerability detection: Are we there yet? *arXiv preprint arXiv:2009.07235*, 2020.

[7] Xinyun Chen, Chang Liu, and Dawn Song. Tree-to-tree neural networks for program translation. pages 2547–2557. Advances in Neural Information Processing Systems 31, 2018.

[8] Zimin Chen, Steve James Kommrusch, Michele Tufano, Louis-Noël Pouchet, Denys Poshyvanyk, and Martin Monperrus. Sequencer: Sequence-to-sequence learning for end-to-end program repair. *IEEE Transactions on Software Engineering(TSE)*, 2019.

[9] Jacob Devlin, Ming-Wei Chang, Kenton Lee, and Kristina Toutanova. BERT: Pre-training of deep bidirectional transformers for language understanding. pages 4171–4186, Minneapolis, Minnesota, June 2019. Association for Computational Linguistics.

[10] Mehdi Drissi, Olivia Watkins, Aditya Khant, Vivaswat Ojha, Pedro Sandoval, Rakia Segev, Eric Weiner, and Robert Keller. Program language translation using a grammar-driven tree-to-tree model. *ICML workshop Neural Abstract Machines Program Induction*, 2018.

[11] Zhangyin Feng, Daya Guo, Duyu Tang, Nan Duan, Xiaocheng Feng, Ming Gong, Linjun Shou, Bing Qin, Ting Liu, Daxin Jiang, et al. Codebert: A pre-trained model for programming and natural languages. *Findings of Empirical Methods in Natural Language Processing (EMNLP-findings)*, 2020.

[12] Daya Guo, Shuo Ren, Shuai Lu, Zhangyin Feng, Duyu Tang, Shujie Liu, Long Zhou, Nan Duan, Jian Yin, Daxin Jiang, et al. Graphcodebert: Pre-training code representations with data flow. *arXiv preprint arXiv:2009.08366*, 2020.

[13] Jacob Harer, Chris Reale, and Peter Chin. Tree-transformer: A transformer-based method for correction of tree-structured data. *Empirical Methods in Natural Language Processing (EMNLP)*, 2019.

[14] Vincent J. Hellendoorn and Premkumar Devanbu. Are deep neural networks the best choice for modeling source code? pages 763–773, New York, NY, USA, 2017. Proceedings of the 2017 11th Joint Meeting on Foundations of Software Engineering (FSE).

[15] Xing Hu, Ge Li, Xin Xia, David Lo, and Zhi Jin. Deep code comment generation. page 200–210, New York, NY, USA, 2018. Proceedings of the 26th Conference on Program Comprehension (ICPC).

[16] Srinivasan Iyer, Ioannis Konstas, Alvin Cheung, and Luke Zettlemoyer. Summarizing source code using a neural attention model. pages 2073–2083, Berlin, Germany, August 2016. Association for Computational Linguistics.

[17] Jian Li, Yue Wang, Michael R Lyu, and Irwin King. Code completion with neural attention and pointer networks. *Empirical Methods in Natural Language Processing*, 2017.

[18] Zhen Li, Deqing Zou, Shouhuai Xu, Xinyu Ou, Hai Jin, Sujuan Wang, Zhijun Deng, and Yuyi Zhong. Vuldeepecker: A deep learning-based system for vulnerability detection. *Network and Distributed Systems Security (NDSS)*, 2018.

[19] Yinhan Liu, Myle Ott, Naman Goyal, Jingfei Du, Mandar Joshi, Danqi Chen, Omer Levy, Mike Lewis, Luke Zettlemoyer, and Veselin Stoyanov. Roberta: A robustly optimized bert pretraining approach. *Under Review on ICLR*, 2020.

[20] Thibaud Lutellier, Hung Viet Pham, Lawrence Pang, Yitong Li, Moshi Wei, and Lin Tan. Coconut: combining context-aware neural translation models using ensemble for program repair. pages 101–114. Proceedings of the 29th ACM SIGSOFT International Symposium on Software Testing and Analysis (ISSTA), 2020.

[21] Dana Movshovitz-Attias and William W. Cohen. Natural language models for predicting programming comments. pages 35–40, Sofia, Bulgaria, August 2013. Proceedings of the 51st Annual Meeting of the Association for Computational Linguistics (ACL).

[22] Matthew Peters, Mark Neumann, Mohit Iyyer, Matt Gardner, Christopher Clark, Kenton Lee, and Luke Zettlemoyer. Deep contextualized word representations. *Proceedings of the 2018 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, Volume 1 (Long Papers) (ACL)*, pages 2227–2237, June 2018.

[23] Baishakhi Ray, Vincent Hellendoorn, Saheel Godhane, Zhaopeng Tu, Alberto Bacchelli, and Premkumar Devanbu. On the" naturalness" of buggy code. pages 428–439. IEEE, 2016 IEEE/ACM 38th International Conference on Software Engineering (ICSE), 2016.

[24] Rebecca Russell, Louis Kim, Lei Hamilton, Tomo Lazovich, Jacob Harer, Onur Ozdemir, Paul Ellingwood, and Marc McConley. Automated vulnerability detection in source code using deep representation learning. pages 757–762. IEEE, 2018 17th IEEE International Conference on Machine Learning and Applications (ICMLA), 2018.

[25] Wenhan Wang, Ge Li, Bo Ma, Xin Xia, and Zhi Jin. Detecting code clones with graph neural network and flow-augmented abstract syntax tree. pages 261–271. IEEE, 2020 IEEE 27th International Conference on Software Analysis, Evolution and Reengineering (SANER), 2020.

[26] Haoran Xu, Shuhui Fan, Yongjun Wang, Zhijian Huang, Hongzuo Xu, and Peidai Xie. Tree2tree structural language modeling for compiler fuzzing. pages 563–578. Springer, International Conference on Algorithms and Architectures for Parallel Processing, 2020.

[27] Hao Yu, Wing Lam, Long Chen, Ge Li, Tao Xie, and Qianxiang Wang. Neural detection of semantic code clones via tree-based convolution. pages 70–80. IEEE, 2019 IEEE/ACM 27th International Conference on Program Comprehension (ICPC), 2019.

[28] Jian Zhang, Xu Wang, Hongyu Zhang, Hailong Sun, and Xudong Liu. Retrieval-based neural source code summarization. IEEE, Proceedings of the 42nd International Conference on Software Engineering (ICSE), 2020.

[29] Jian Zhang, Xu Wang, Hongyu Zhang, Hailong Sun, Kaixuan Wang, and Xudong Liu. A novel neural source code representation based on abstract syntax tree. page 783–794. International Conference on Software Engineering (ICSE), 2019.

[30] Yaqin Zhou, Shangqing Liu, Jingkai Siow, Xiaoning Du, and Yang Liu. Devign: Effective vulnerability identification by learning comprehensive program semantics via graph neural networks. pages 10197–10207. Advances in Neural Information Processing Systems, 2019.