Intelligent Software Engineering: Synergy between AI and Software Engineering*

Tao Xie

University of Illinois at Urbana-Champaign, Urbana, IL 61801, USA
taoxie@illinois.edu

Abstract. As an example of exploiting the synergy between AI and software engineering, the field of intelligent software engineering has emerged with various advances in recent years. Such field broadly addresses issues on *intelligent* [software engineering] and [*intelligence software*] engineering. The former, *intelligent* [software engineering], focuses on instilling intelligence in approaches developed to address various software engineering tasks to accomplish high effectiveness and efficiency. The latter, [*intelligence software*] engineering, focuses on addressing various software engineering tasks for intelligence software, e.g., AI software. In this paper, we discuss recent research and future directions in the field of intelligent software engineering.

Keywords: Intelligent software engineering.

1 Introduction

As an example of exploiting the synergy between AI and software engineering, the field of intelligent software engineering [31] has emerged with various advances in recent years. Such field broadly addresses issues on *intelligent* [software engineering] and [*intelligence software*] engineering. The former, *intelligent* [software engineering], focuses on instilling intelligence in approaches developed to address various software engineering tasks to accomplish high effectiveness and efficiency. The latter, [*intelligence software*] engineering, focuses on addressing various software engineering tasks for intelligence software, e.g., AI software. Indeed, the preceding two aspects can overlap when instilling intelligence in approaches developed to address various software engineering tasks for intelligence software. By nature, the field of intelligent software engineering is a research field spanning at least the research communities of software engineering and AI.

2 Instilling Intelligence in Software Engineering

Applying or adapting AI technologies to address various software engineering tasks [13] has been actively pursued by researchers from the software engineering

* This work was supported in part by National Science Foundation under grants no. CNS-1513939 and CNS1564274, and a grant from the ZJUI Research Program.

research community, e.g., machine learning for software engineering [18, 1, 38] and natural language processing for software engineering [29, 39, 20, 21, 30, 40], and also by researchers from the AI research community in recent years [2] partly due to the increasing popularity of deep learning [24]. Much of such previous work has been on automating as much as possible to address a specific software engineering task such as programming and testing. But as pointed out by various AI researchers [17, 14], AI technologies typically enhance or augment human, instead of replacing human.

In future work, we envision that intelligence can be instilled into approaches for software engineering tasks in the following two example ways as starting points.

Natural language interfacing. Natural language conversations between a human and a machine can be traced back to the Turing test [28], proposed by Alan Turing in 1950, as a test for a machine to exhibit intelligent behaviors indistinguishable from a human's. Natural-language-based chatbots have been increasingly developed and deployed for various domains: virtual assistants (such as Apple Siri, Google Assistant, Amazon Alexa, Microsoft Cortana, Samsung Bixby), customer services, social media (such as Facebook Messenger chatbots). Very recently, exploring the use of chatbots in software engineering has been started [26, 15, 5, 6, 12, 3]. Beyond chatbots or conversational natural language interfacing, natural language interfacing will play an increasingly important and popular role in software development environments [9], due to its benefits of improving developer productivity.

Continuous learning. Machine learning has been increasingly applied or adapted for various software engineering tasks since at least early 2000 [32, 2]; in the past several years, deep learning [24] has been applied on software engineering problems (e.g., [10, 11, 35]). Such direction's increasing popularity is partly thanks to the availability of rich data (being either explicitly or implicitly labeled in one way or another) in software repositories [32] along with the advances in machine learning especially deep learning [24] in recent years. Beyond applying machine learning only once or occasionally, software engineering tools are in need of gaining the continuous-learning capability: when the tools are applied in software engineering practices, the tools continuously learn to get more and more intelligent and capable.

3 Software Engineering for Intelligence Software

In recent decades, Artificial Intelligence (AI) has emerged as a technical pillar underlying modern-day solutions to increasingly important tasks in daily life and work. The impacted settings range from smartphones carried in one's pocket to transportation vehicles. Artificial Intelligence (AI) solutions are typically in the form of software. Thus, intelligence software is naturally amenable to software engineering issues such as dependability [8] including reliability [22, 27] and security [33, 34], etc. Assuring dependability of intelligence software is critical but largely unexplored compared to traditional software.

For example, intelligence software that interacts with users by communicating content (e.g., chatbots, image-tagging) does so within social environments constrained by social norms [7]. For such intelligence software to meet the users' expectation, it must comply with accepted social norms. However, determining what are accepted social norms can vary greatly by culture and environment [7]. Recent AI-based solutions, such as Microsoft's intelligent chatbot Tay [16] and Google Photos app's imagine-tagging feature [4], received negative feedback because their behavior lied outside accepted social norms. In addition, formulating failure conditions and monitoring such conditions at runtime may not be acceptable for intelligence software such as robotic software because it would be too late to conduct failure avoidance or recovery actions when such failure conditions are detected [23]. Generally formulating proper requirements for intelligence software remains a challenge for the research community.

In addition, intelligence software is known to often suffer from the "no oracle problem" [19, 36, 25, 37]. For example, in supervised learning, a future application-data entry can be labeled (manually or automatically); however, using such labels as the test oracle is not feasible. The reason is that there exists some inaccuracy (i.e., predicting a wrong label) in the learned classification model. This inaccuracy is inherent and sometimes desirable to avoid the overfitting problem (i.e., the classification model performs perfectly on the training data but undesirably in future application data).

References

1. Acharya, M., Xie, T., Pei, J., Xu, J.: Mining API patterns as partial orders from source code: From usage scenarios to specifications. In: Proc. Meeting of the European Software Engineering Conference and the ACM SIGSOFT Symposium on The Foundations of Software Engineering (ESEC-FSE). pp. 25–34 (2007)
2. Allamanis, M., Barr, E.T., Devanbu, P., Sutton, C.: A survey of machine learning for big code and naturalness. In: eprint arXiv:1709.06182 (September 2017)
3. Balog, M., Gaunt, A.L., Brockschmidt, M., Nowozin, S., Tarlow, D.: DeepCoder: Learning to write programs. In: Proc. International Conference on Learning Representations (ICLR) (2017)
4. Barr, A.: Google mistakenly tags black people as gorillas, showing limits of algorithms. The Wall Street Journal (2015), http://blogs.wsj.com/digits/2015/07/01/google-mistakenly-tags-black-people-as-gorillas-showing-limits-of-algorithms/
5. Beschastnikh, I., Lungu, M.F., Zhuang, Y.: Accelerating software engineering research adoption with analysis bots. In: Proc. International Conference on Software Engineering (ICSE), New Ideas and Emerging Results Track. pp. 35–38 (2017)
6. Bieliauskas, S., Schreiber, A.: A conversational user interface for software visualization. In: Proc. IEEE Working Conference on Software Visualization (VISSOFT). pp. 139–143 (2017)
7. Coleman, J.: Foundations of Social Theory. Belknap Press Series, Belknap Press of Harvard University Press (1990)
8. Committee on Technology National Science and Technology Council and Penny Hill Press: Preparing for the Future of Artificial Intelligence. CreateSpace Independent Publishing Platform, USA (2016)

9. Ernst, M.D.: Natural language is a programming language: Applying natural language processing to software development. In: Proc. the 2nd Summit oN Advances in Programming Languages (SNAPL). pp. 4:1–4:14 (2017)
10. Gu, X., Zhang, H., Zhang, D., Kim, S.: Deep API learning. In: Proc. ACM SIGSOFT International Symposium on Foundations of Software Engineering (FSE). pp. 631–642 (2016)
11. Gu, X., Zhang, H., Zhang, D., Kim, S.: DeepAM: Migrate APIs with multi-modal sequence to sequence learning. In: Proc. International Joint Conference on Artificial Intelligence (IJCAI). pp. 3675–3681 (2017)
12. Gupta, R., Pal, S., Kanade, A., Shevade, S.: DeepFix: Fixing common C language errors by deep learning. In: Proc. National Conference on Artificial Intelligence (AAAI) (2017)
13. Harman, M.: The role of artificial intelligence in software engineering. In: Proc. International Workshop on Realizing AI Synergies in Software Engineering (RAISE). pp. 1–6 (2012)
14. Jordan, M.: Artificial intelligence -the revolution hasn't happened yet (April 2018), https://medium.com/@mijordan3/artificial-intelligence-the-revolution-hasnt-happened-yet-5e1d5812e1e7
15. Lebeuf, C., Storey, M.D., Zagalsky, A.: How software developers mitigate collaboration friction with chatbots. CoRR **abs/1702.07011** (2017), http://arxiv.org/abs/1702.07011
16. Leetaru, K.: How Twitter corrupted Microsoft's Tay: A crash course in the dangers of AI in the real world. Forbes (2016), https://www.forbes.com/sites/kalevleetaru/2016/03/24/how-twitter-corrupted-microsofts-tay-a-crash-course-in-the-dangers-of-ai-in-the-real-world/
17. Li, F.F.: How to make A.I. that's good for people (March 2018), https://www.nytimes.com/2018/03/07/opinion/artificial-intelligence-human.html
18. Michail, A., Xie, T.: Helping users avoid bugs in GUI applications. In: Proc. International Conference on Software Engineering (ICSE). pp. 107–116 (2005)
19. Murphy, C., Kaiser, G.E.: Improving the dependability of machine learning applications. Tech. Rep. CUCS-049-, Department of Computer Science, Columbia University (2008)
20. Pandita, R., Xiao, X., Yang, W., Enck, W., Xie, T.: WHYPER: Towards automating risk assessment of mobile applications. In: Proc. USENIX Conference on Security (SEC). pp. 527–542 (2013)
21. Pandita, R., Xiao, X., Zhong, H., Xie, T., Oney, S., Paradkar, A.: Inferring method specifications from natural language API descriptions. In: Proc. International Conference on Software Engineering (ICSE). pp. 815–825 (2012)
22. Pei, K., Cao, Y., Yang, J., Jana, S.: DeepXplore: Automated whitebox testing of deep learning systems. In: Proc. Symposium on Operating Systems Principles (SOSP). pp. 1–18 (2017)
23. Qin, Y., Xie, T., Xu, C., Astorga, A., Lu, J.: CoMID: Context-based multi-invariant detection for monitoring cyber-physical software. CoRR **abs/1807.02282** (2018), https://arxiv.org/abs/1807.02282
24. Schmidhuber, J.: Deep learning in neural networks. Neural Netw. **61**(C), 85–117 (Jan 2015)
25. Srisakaokul, S., Wu, Z., Astorga, A., Alebiosu, O., Xie, T.: Multiple-implementation testing of supervised learning software. In: Proc. AAAI-18 Workshop on Engineering Dependable and Secure Machine Learning Systems (EDSMLS) (2018)

26. Storey, M.D., Zagalsky, A.: Disrupting developer productivity one bot at a time. In: Proc. ACM SIGSOFT International Symposium on Foundations of Software Engineering (FSE). pp. 928–931 (2016)
27. Tian, Y., Pei, K., Jana, S., Ray, B.: DeepTest: Automated testing of deep-neural-network-driven autonomous cars. In: Proc. International Conference on Software Engineering (ICSE). pp. 303–314 (2018)
28. Turing, A.M.: Computing machinery and intelligence (1950), one of the most influential papers in the history of the cognitive sciences: http://cogsci.umn.edu/millennium/final.html
29. Wang, X., Zhang, L., Xie, T., Anvik, J., Sun, J.: An approach to detecting duplicate bug reports using natural language and execution information. In: Proc. International Conference on Software Engineering (ICSE). pp. 461–470 (2008)
30. Xiao, X., Paradkar, A., Thummalapenta, S., Xie, T.: Automated extraction of security policies from natural-language software documents. In: Proc. ACM SIGSOFT International Symposium on the Foundations of Software Engineering (FSE). pp. 12:1–12:11 (2012)
31. Xie, T.: Intelligent software engineering: Synergy between AI and software engineering. In: Proc. Innovations in Software Engineering Conference (ISEC). p. 1:1 (2018)
32. Xie, T., Thummalapenta, S., Lo, D., Liu, C.: Data mining for software engineering. Computer **42**(8), 55–62 (Aug 2009)
33. Yang, W., Kong, D., Xie, T., Gunter, C.A.: Malware detection in adversarial settings: Exploiting feature evolutions and confusions in Android apps. In: Proc. Annual Computer Security Applications Conference (ACSAC). pp. 288–302 (2017)
34. Yang, W., Xie, T.: Telemade: A testing framework for learning-based malware detection systems. In: Proc. AAAI-18 Workshop on Engineering Dependable and Secure Machine Learning Systems (EDSMLS) (2018)
35. Yin, P., Neubig, G.: A syntactic neural model for general-purpose code generation. In: Proc. Annual Meeting of the Association for Computational Linguistics (ACL) (2017)
36. Zheng, W., Ma, H., Lyu, M.R., Xie, T., King, I.: Mining test oracles of web search engines. In: Proc. IEEE/ACM International Conference on Automated Software Engineering (ASE). pp. 408–411 (2011)
37. Zheng, W., Wang, W., Liu, D., Zhang, C., Zeng, Q., Deng, Y., Yang, W., Xie, T.: Oracle-free detection of translation issue for neural machine translation. CoRR **abs/1807.02340** (2018), https://arxiv.org/abs/1807.02340
38. Zhong, H., Xie, T., Zhang, L., Pei, J., Mei, H.: MAPO: Mining and recommending api usage patterns. In: Proc. European Conference on Object-Oriented Programming (ECOOP). pp. 318–343 (2009)
39. Zhong, H., Zhang, L., Xie, T., Mei, H.: Inferring resource specifications from natural language API documentation. In: Proc. IEEE/ACM International Conference on Automated Software Engineering (ASE). pp. 307–318 (2009)
40. Zhong, Z., Guo, J., Yang, W., Xie, T., Lou, J.G., Liu, T., Zhang, D.: Generating regular expressions from natural language specifications: Are we there yet? In: Proc. Workshop on NLP for Software Engineering (NL4SE) (2018)