Programming in the Age of Intelligent Machines

Breandan Considine

August 10, 2021

1 Introduction

Since the invention of modern computers in the mid 20th century, computer programming has undergone a number of paradigm shifts. From the rise of functional programming, to dynamic, object-oriented, and type-level programming, to the availability of myriad tools and frameworks – its practitioners have witnessed a veritable Renaissance in the art of computer programming. With each of these paradigm shifts, programmers have realized new conceptual frameworks for reasoning and expressing their ideas more clearly and concisely.

Over the last few years, another paradigm shift has been set in motion, with significant implications for how we think about and write programs in the coming century. By most measures, computers have grown steadily more intelligent and capable of assisting programmers with mentally taxing chores. For example, intelligent programming tools (IPTs) powered by neural language models have this year helped over 10 million unique human beings program computers. As IPTs help digitally illiterate communities to discover their innate aptitude for computer programming, this population will continue to rise.

The art of computer programming is a uniquely creative exercise among the range of human activities. It channels our innate linguistic, logical, imaginative, and social abilities to bring abstract ideas into reality, and ultimately, gives humans the freedom to create new realities of their own design. In collaboration with fellow humans and the increasing participation of IPTs, vast and elaborate virtual worlds have been manufactured, where the majority of humankind now chooses to spend their daily lives. As these new digital frontiers offer increasingly compelling relocation benefits, their population too will continue to grow.

Today IPTs share an equal role in shaping many aspects of computer programming, from knowledge discovery to API design, and program synthesis to validation and verification. However, this balance is shifting beneath our feet. Once its creators, programmers are now primarily consumers of information provided by an IPT, and increasingly rely on them to perform their daily work. With the unique opportunities and risks their partnership presents, what division of labor should exist between humans and their programming assistants? This is the question we set out to understand in the following literature review.

2 Neural Language Models for Source Code

Programming researchers have long held an interest in using intelligent tools to help write programs [2]. Due to fundamental limitations in data and processing power, many of these ideas could not be fully realized until recently, thanks in part to the development of transformer-based models [4] capable of learning more complex languages with longer-range dependencies. Other developments, such as more efficient representations and learning algorithms [1] have contributed to the renewed interest in this research direction.

Since then, an enormous number of transformer-based models have been published and researchers have made rapid progress in industrial transfer, where this technology is now trained and deployed on millions of programmers worldwide [3]. By training on open source data collected from GitHub, these models are capable of inferring the programmer's intent and completing long fragments of source code given some contextual information.

3 Knowledge Discovery and Neural Code Search

4 Computer-Aided Reasoning Tools

5 Automatic and Synthetic Programming

6 Future Directions of Computer Programming

7 Conclusion

References

[1] Miltiadis Allamanis, Earl T Barr, Premkumar Devanbu, and Charles Sutton. A survey of machine learning for big code and naturalness. *ACM Computing Surveys (CSUR)*, 51(4):1–37, 2018.

[2] M Bras and Y Toussaint. Artificial intelligence tools for software engineering: Processing natural language requirements. *WIT Transactions on Information and Communication Technologies*, 2, 1993.

[3] Mark Chen, Jerry Tworek, Heewoo Jun, Qiming Yuan, Henrique Ponde, Jared Kaplan, Harri Edwards, Yura Burda, Nicholas Joseph, Greg Brockman, et al. Evaluating large language models trained on code. *arXiv preprint arXiv:2107.03374*, 2021.

[4] Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N Gomez, Łukasz Kaiser, and Illia Polosukhin. Attention is all you need. In *Advances in neural information processing systems*, pages 5998–6008, 2017.