Applied AI Matters
AI4Code: Applying Artificial Intelligence to Source Code

Kartik Talamadupula (IBM Research AI; krtalamad@us.ibm.com)

DOI: 10.1145/3465074.3465080

Introduction

The marriage of Artificial Intelligence (AI) techniques to problems surrounding the generation, maintenance, and use of source code has come to the fore in recent years as an important AI application area[1]. A large chunk of this recent attention can be attributed to contemporaneous advancements in Natural Language Processing (NLP) techniques and sub-fields. The *naturalness hypothesis*, which states that "software is a form of human communication" and that code exhibits patterns that are similar to (human) natural languages (Devanbu, 2015; Hindle, Barr, Gabel, Su, & Devanbu, 2016), has allowed for the application of many of these NLP advances to code-centric usecases. This development has contributed to a spate of work in the community – much of it captured in a survey by Allamanis, Barr, Devanbu, and Sutton (2018) that focuses on classifying these approaches by the type of probabilistic model applied to source code.

This increase in the variety of AI techniques applied to source code has found various manifestations in the industry at large. Code and software form the backbone that underpins almost all modern technical advancements: it is thus natural that breakthroughs in this area should reflect in the emergence of real world deployments.

AI4Code: Industrial Applications

There are several characterizations and groupings that can be made when considering applications of AI4Code. One that has already been discussed is predicated on the kinds of probabilistic models of code that are generated and exploited. Another is in terms of whether AI techniques are addressing code usecases (AI4Code); if code is being used to make AI problems easier to solve (Code4AI); or if AI/ML techniques are themselves being used to improve AI tools and lifecycles (AI4AI, or AutoAI/AutoML). Then there are classifications that are based on the specific location in the development-devops cycle where the AI techniques are being injected: this can be any stage starting from requirements gathering, through code generation and documentation, translation, testing, execution, and deployment. A final classification is in terms of the user role that is targeted by these AI4Code manifestations: some might target developers, while others target devops personas.

In the following, we present a non-exhaustive list of some recent AI4Code tools from a wide cross-section of industrial and applied research settings in order to introduce readers to the variety of AI applications in this space.

- **AutoAI** is a stream of work that applies AI techniques to automate machine learning and data science pipelines; recent work has looked at the issues inherent with humans in the loop in such end-to-end lifecycles (Wang et al., 2020).
- **CLAI**[2] is an open-source project that brings AI advances to the command line to automate and ease developer and devops usecases (Agarwal, Barroso, et al., 2020).
- **CodeGuru**[3] is a tool that provides developer oversight of code, particularly with an eye towards efficiency and cost.
- **CriticalHop**[4] is a commercial AI planning engine that seeks to help users minimize Kubernetes and cloud deployment issues.
- **DeepCode**[5] is an AI enabled code review engine that performs semantic code analysis to find critical issues and vulnerabilities.

[1] This report builds on the content of a panel at the NeurIPS 2020 Industrial Expo on *AI4Code at IBM + RedHat*, hosted by the author.

[2] https://github.com/ibm/clai
[3] https://aws.amazon.com/codeguru/
[4] https://www.criticalhop.com/
[5] https://www.deepcode.ai/

- **Graph4Code**[6] is a knowledge graph over code from Python programs that captures the semantics of that code.
- **IntelliCode**[7] is an IDE plugin that tries to enable faster and more efficient coding for developers.
- **Kite**[8] is an AI-powered code completion assistant that aims to minimize developer keystrokes.
- **ModelOps** is a cloud-based platform for end-to-end development and lifecycle management of AI applications (Hummer et al., 2019).
- **Mono2Micro**[9] uses AI techniques to offer recommendations to refactor monolithic code into microservices.
- **Thoth**[10] uses AI techniques to analyze and recommend software stacks for AI applications during deployment.

Code Translation with Neural Models: A Human in the Loop Case Study

While the transfer of AI techniques into industrial tools for code has kept up a brisk pace, there is still much work to be done in terms of evaluating the impact of all these groundbreaking techniques and enriched tools. Specifically, the effect that these tools have on developers – who are the users at the center of this AI4Code revolution – remains to be fully measured. There have been nascent efforts in this space, e.g. the HAI-GEN workshop series[11] at the ACM Intelligent User Interfaces (IUI) conference; and Xu, Vasilescu, and Neubig (2021)'s work.

One recent area of interest under the AI4Code umbrella has been the problem of code translation – that is, automatically translating source code in one language to another. Code translation has applications in many important scenarios, including the modernization of legacy code that runs critical infrastructure and applications in industries such as finance, travel, and government. There have been a number of AI4Code approaches (Nguyen, Nguyen, & Nguyen, 2014; Oda et al., 2015) that seek to cast the code translation problem as a special case of the more general *machine translation* task, which entails automatic translation between two (human) natural languages. Recently, unsupervised machine translation techniques have been applied to the code translation task, with great success. This is exemplified by the TransCoder (Roziere, Lachaux, Chanussot, & Lample, 2020) system, which trains a fully unsupervised neural transcompiler to translate functions between Java, C++, and Python.

The TransCoder model, which is a sequence to sequence (seq2seq) model (Sutskever, Vinyals, & Le, 2014), generates tokens at inference time that together make up the translation of a given input (source) function. The model is able to generate multiple translations using beam search decoding; this information can be used to compute token-level confidence scores for each token produced by the model. These confidence scores can then be shown to the human end-user in the absence of ground truth about a specific translation.

However, this raises an interesting issue: what is a human user to infer from these confidences? Apart from being able to order them ordinally, there is no correlation between the task at hand (code translation) and the output of the model. Agarwal, Talamadupula, et al. (2020) address this problem, and seek to *anchor* the confidence scores by correlating them to linter errors that are generated by the translated code. The thesis underlying their work is that human users – particularly those who are going to interact with and ultimately use the output of the AI system – need to sufficiently ground their understanding of that output to some intermediate representation from their domain of expertise. A detailed analysis of a user study on this usecase is presented in Weisz et al. (2021).

Conclusion

In this article, we sought to briefly introduce and explore the area of *AI4Code* – i.e., the application of AI techniques to usecases revolving around source code. We showed that

[6] https://wala.github.io/graph4code/
[7] https://visualstudio.microsoft.com/services/intellicode/
[8] https://www.kite.com/
[9] http://ibm.biz/Mono2Micro
[10] https://thoth-station.ninja
[11] https://hai-gen2021.github.io/

in recent years, there has been an explosion of work in this area thanks to the naturalness hypothesis that draws a direct link between source code artefacts and the recent leaps in NLP that have occurred on human natural languages. We discussed the manifestation of some of these techniques in industrial and applied research scenarios, and provided a broad list of AI4Code deployments. This was followed by a deep dive into one specific effort centered around code translation; and an examination of one human in the loop issue around the deployment of this usecase. AI4Code remains a very exciting and fast-moving application area, and promises many breakthroughs in the days to come.

References

Agarwal, M., Barroso, J. J., Chakraborti, T., Dow, E. M., Fadnis, K., Godoy, B., & Talamadupula, K. (2020). CLAI: A Platform for AI Skills on the Command Line. *arXiv preprint arXiv:2002.00762*.

Agarwal, M., Talamadupula, K., Houde, S., Martinez, F., Muller, M., Richards, J., ... Weisz, J. D. (2020). Quality Estimation & Interpretability for Code Translation. *Computer Assisted Programming (CAP) Workshop at NeurIPS 2020*.

Allamanis, M., Barr, E. T., Devanbu, P., & Sutton, C. (2018). A survey of machine learning for big code and naturalness. *ACM Computing Surveys (CSUR)*, *51*(4), 1–37.

Devanbu, P. (2015). New initiative: The naturalness of software. In *2015 IEEE/ACM 37th IEEE International Conference on Software Engineering* (Vol. 2, pp. 543–546).

Hindle, A., Barr, E. T., Gabel, M., Su, Z., & Devanbu, P. (2016). On the naturalness of software. *Communications of the ACM*, *59*(5), 122–131.

Hummer, W., Muthusamy, V., Rausch, T., Dube, P., El Maghraoui, K., Murthi, A., & Oum, P. (2019). Modelops: Cloud-based lifecycle management for reliable and trusted ai. In *2019 IEEE International Conference on Cloud Engineering (IC2E)* (pp. 113–120).

Nguyen, A. T., Nguyen, T. T., & Nguyen, T. N. (2014). Migrating code with statistical machine translation. In *Companion Proceedings of the 36th International Conference on Software Engineering* (pp. 544–547).

Oda, Y., Fudaba, H., Neubig, G., Hata, H., Sakti, S., Toda, T., & Nakamura, S. (2015). Learning to generate pseudo-code from source code using statistical machine translation (t). In *2015 30th IEEE/ACM International Conference on Automated Software Engineering (ASE)* (pp. 574–584).

Roziere, B., Lachaux, M.-A., Chanussot, L., & Lample, G. (2020). Unsupervised translation of programming languages. *Advances in Neural Information Processing Systems*, *33*.

Sutskever, I., Vinyals, O., & Le, Q. V. (2014). Sequence to sequence learning with neural networks. *arXiv preprint arXiv:1409.3215*.

Wang, D., Ram, P., Weidele, D. K. I., Liu, S., Muller, M., Weisz, J. D., ... others (2020). Autoai: Automating the end-to-end ai lifecycle with humans-in-the-loop. In *Proceedings of the 25th International Conference on Intelligent User Interfaces Companion* (pp. 77–78).

Weisz, J. D., Muller, M., Houde, S., Richards, J., Ross, S. I., Martinez, F., ... Talamadupula, K. (2021). Perfection Not Required? Human-AI Partnerships in Code Translation. In *26th Annual Conference on Intelligent User Interfaces (IUI).*

Xu, F. F., Vasilescu, B., & Neubig, G. (2021). In-IDE Code Generation from Natural Language: Promise and Challenges. *arXiv preprint arXiv:2101.11149*.

Kartik Talamadupula is a Research Scientist at IBM Research AI, and a AAAI Senior Member. He also serves as the SIGAI Applied AI Officer. His recent research interests are in applying AI to Business Automation, and in studying the implications of human users interacting with Generative AI models. Kartik can be reached at krtalamad@us.ibm.com.