Generation Probabilities are Not Enough: Improving Error Highlighting for AI Code Suggestions

Helena Vasconcelos *
Stanford University
helenav@cs.stanford.edu

Gagan Bansal†
Microsoft Research
gaganbansal@microsoft.com

Adam Fourney†
Microsoft Research
adamfo@microsoft.com

Q. Vera Liao†
Microsoft Research
veraliao@microsoft.com

Jennifer Wortman Vaughan†
Microsoft Research
jenn@microsoft.com

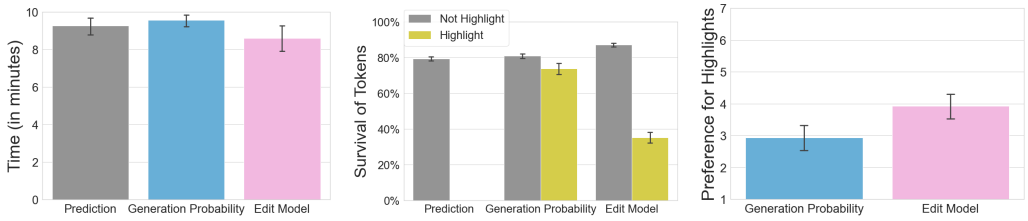
1 Introduction

Large-scale generative models are increasingly being used in tooling applications. As one prominent example, code generation models, such as Copilot [8], CodeWhisperer [1], and AlphaCode [7], recommend code completions within an IDE to help programmers author software. However, since these models are imperfect, their erroneous recommendations can introduce bugs or even security vulnerabilities into a code base if not overridden by a human user [14]. In order to override such errors, users must first detect them. This can be challenging as even experts may be susceptible to automation bias and automation-induced complacency [13, 17]. To help users detect errors in medical [5, 11], legal [2, 9], and other high-stakes domains, conveying AI uncertainty and providing explanations has become of paramount importance [4, 3]. However, prior scenarios focus on decision support (e.g., a single classification or diagnosis), and it is not clear how to translate these strategies to generative scenarios where every generation may include dozens or hundreds of small decisions (e.g., each token of recommended code). Likewise, it is unclear how best to convey the uncertainty of generative models to human operators or if doing so will positively impact human-AI collaboration.

To make progress on these questions, through a mixed-methods, preregistered (https://osf.io/tymah) study with $N = 30$ participants, we explore the value of token-level highlighting in a code generation scenario. Similar to in-line spell-check in text editors, highlighted tokens are meant to draw operator attention to regions of the code that would benefit most from human oversight. In our study, we explore two possible highlighting strategies, together with a baseline without highlights. Our first strategy highlights tokens with lowest probabilities, as output directly from the underlying language model. The intuition is that low-probability sequences are non-conventional, and therefore may indicate an error. This highlighting strategy has been proposed in past work [15] and is implemented in OpenAI's online "Playground" interface [12]. Conversely, our second highlighting strategy directly predicts the need for intervention by predicting which tokens of a suggestion are likely to be edited. For this paper, we learn an edit model for a closed-world set of programming tasks, but argue that there are clear paths to generalize the approach by learning from existing large-scale data, including telemetry. **From our study, we find the edit model strategy results in significantly faster task completion time, significantly more localized edits, and was strongly preferred by participants.**

*Work done while at Microsoft Research.
†Equal contribution last authors; ordered by last name.

36th Conference on Neural Information Processing Systems (NeurIPS 2022).



(a) (b) (c)

Figure 1: **(a):** Time spent on task (max. 10 mins). **(b):** Survival of tokens in each condition, separated by whether the token was highlighted or not. **(c):** Self-reported preference for highlights.

2 Study and Results

We conducted an interview study with 30 participants. All were employees of a large technology company located in the United States and have experience coding in Python. Participation was voluntary and participants were paid $50. Interviews lasted approximately one hour. Participants were asked to complete three coding tasks, selected from Leetcode's "easy" setting [10], with three AI-powered code completion tools. In actuality, all three tools used OpenAI's Davinci-002 Codex model [6], and differed only in how completions were presented: One tool showed only the generated code completion (**Prediction only**). Another showed the completion with highlights on tokens that were most uncertain (threshold at 71%) as output by Codex (**Generation probability**). The final tool highlighted tokens that were most likely to be edited (threshold at 66%) according to our edit model (**Edit model**). The edit model was built using data collected from nine coders in a preliminary data collection phase: Coders were asked to edit the Codex output until the task was completed properly. We chose the highlighting thresholds such that the total number of highlights shown across all three tasks for each condition were equivalent. The order of tasks as well as assignments of tools to tasks were randomized. Participants were able to run their code for debugging, and also run a set of provided unit tests. Once participants were satisfied with their solution, or after a limit of 10 minutes, participants were asked a series of questions rating their experience.

Our results show that token-level highlighting meaningfully impacts user behavior, and further that the choice of highlighting strategy yields critical differences. For instance, participants were fastest in the edit highlighting condition ($\mu = 8.59$ minutes), and slowest in the generation probability condition ($\mu = 9.61$ minutes), with the prediction only condition occupying the middle-ground ($\mu = 9.27$ minutes) (Fig 1a). The difference between the two highlighting conditions is highly significant ($p = 0.003$), while the difference between the edit condition and the prediction only condition shows a promising trend in this direction ($p = 0.06$). Echoing this finding, our results also indicate that our edit model steers people towards making more precise edits (Fig 1b). Non-highlighted tokens are significantly more likely to remain untouched by the participant in the edit model condition ($\mu_{survives} = 0.87$), than in both the generation probability condition ($\mu_{survives} = 0.81$, $p < .0001$), and the prediction only condition ($\mu_{survives} = 0.79$, $p < .0001$). Conversely, tokens that *are* highlighted in the edit model condition survive significantly *less* often ($\mu_{survives} = 0.35$) than tokens highlighted in the generation probability condition ($\mu_{survives} = 0.74$, $p < .0001$), meaning that the edit model is a stronger signal of what will be edited by people. Finally, this preference is echoed in participants' subjective ratings to the following 7-point Likert items: *"I found the AI's highlights helpful in determining what to edit"*; *"I would be willing to pay to access the AI's highlights"*; and *"I found the AI's highlights distracting"* (reverse coded). Here we find an average response of 3.94 for the edit conditions vs 2.88 for generation probability condition (Fig 1c), and this difference is again significant with $p = 0.001$.

3 Conclusion and Future Work

Together our findings demonstrate that highlighting appropriate tokens in generated code can meaningfully impact and improve operator behavior, and point a clear path in favor of edit models for this purpose. However, important research questions and challenges remain. First, our work relies on a closed-world edit model, learned on the very AI-generations evaluated in our study. This clearly represents a best-case perfectly-calibrated scenario. It remains to be demonstrated that we can

learn an open-world general-purpose edit model, and that such a model would similarly impact user interaction. Fortunately, systems like GitHub Copilot already consider edits to AI-generations as a form of performance metric [18] and this product-scale data stream could be directly repurposed to this end. Secondly, it remains to be demonstrated that the observed benefits (task completion time, targeted edits, preference), translate to important high-level outcomes such as more accurate human oversight and reduced automation bias. However, past work has shown that reducing the effort needed to interpret model explanations, or expressions of uncertainty, can increase the likelihood of people overriding AI-induced errors [16]. We hope to explore these questions in immediate future work.

References

[1] Amazon Web Services. 2022. *ML-powered coding companion - Amazon CodeWhisperer*. Retrieved September, 2022 from `https://aws.amazon.com/codewhisperer/`

[2] Julia Angwin, Jeff Larson, Surya Mattu, and Lauren Kirchner. 2016. Machine bias: There's software across the country to predict future criminals and it's biased against blacks. (2016).

[3] Gagan Bansal, Tongshuang Wu, Joyce Zhou, Raymond Fok, Besmira Nushi, Ece Kamar, Marco Tulio Ribeiro, and Daniel Weld. 2021. Does the whole exceed its parts? the effect of ai explanations on complementary team performance. In *Proceedings of the 2021 CHI Conference on Human Factors in Computing Systems*. 1–16.

[4] Umang Bhatt, Javier Antorán, Yunfeng Zhang, Q Vera Liao, Prasanna Sattigeri, Riccardo Fogliato, Gabrielle Melançon, Ranganath Krishnan, Jason Stanley, Omesh Tickoo, et al. 2021. Uncertainty as a form of transparency: Measuring, communicating, and using uncertainty. In *Proceedings of the 2021 AAAI/ACM Conference on AI, Ethics, and Society*. 401–413.

[5] Carrie J Cai, Samantha Winter, David Steiner, Lauren Wilcox, and Michael Terry. 2019. " Hello AI": Uncovering the Onboarding Needs of Medical Practitioners for Human-AI Collaborative Decision-Making. *Proceedings of the ACM on Human-computer Interaction* 3, CSCW (2019), 1–24.

[6] Mark Chen, Jerry Tworek, Heewoo Jun, Qiming Yuan, Henrique Ponde de Oliveira Pinto, Jared Kaplan, Harri Edwards, Yuri Burda, Nicholas Joseph, Greg Brockman, Alex Ray, Raul Puri, Gretchen Krueger, Michael Petrov, Heidy Khlaaf, Girish Sastry, Pamela Mishkin, Brooke Chan, Scott Gray, Nick Ryder, Mikhail Pavlov, Alethea Power, Lukasz Kaiser, Mohammad Bavarian, Clemens Winter, Philippe Tillet, Felipe Petroski Such, Dave Cummings, Matthias Plappert, Fotios Chantzis, Elizabeth Barnes, Ariel Herbert-Voss, William Hebgen Guss, Alex Nichol, Alex Paino, Nikolas Tezak, Jie Tang, Igor Babuschkin, Suchir Balaji, Shantanu Jain, William Saunders, Christopher Hesse, Andrew N. Carr, Jan Leike, Josh Achiam, Vedant Misra, Evan Morikawa, Alec Radford, Matthew Knight, Miles Brundage, Mira Murati, Katie Mayer, Peter Welinder, Bob McGrew, Dario Amodei, Sam McCandlish, Ilya Sutskever, and Wojciech Zaremba. 2021. Evaluating Large Language Models Trained on Code. `https://doi.org/10.48550/ARXIV.2107.03374`

[7] DeepMind. 2022. *AlphaCode*. Retrieved September, 2022 from `https://alphacode.deepmind.com/`

[8] GitHub. 2022. *GitHub Copilot - Your AI pair programmer*. Retrieved September, 2022 from `https://github.com/features/copilot/`

[9] Yugo Hayashi and Kosuke Wakabayashi. 2017. Can AI become reliable source to support human decision making in a court scene?. In *Companion of the 2017 ACM Conference on Computer Supported Cooperative Work and Social Computing*. 195–198.

[10] LeetCode. 2015. The world's leading online programming learning platform. `https://leetcode.com/`

[11] Scott M Lundberg, Bala Nair, Monica S Vavilala, Mayumi Horibe, Michael J Eisses, Trevor Adams, David E Liston, Daniel King-Wai Low, Shu-Fang Newman, Jerry Kim, et al. 2018. Explainable machine-learning predictions for the prevention of hypoxaemia during surgery. *Nature biomedical engineering* 2, 10 (2018), 749–760.

[12] OpenAI. 2015. https://beta.openai.com/playground

[13] Raja Parasuraman and Dietrich H Manzey. 2010. Complacency and bias in human use of automation: An attentional integration. *Human factors* 52, 3 (2010), 381–410.

[14] Hammond Pearce, Baleegh Ahmad, Benjamin Tan, Brendan Dolan-Gavitt, and Ramesh Karri. 2022. Asleep at the Keyboard? Assessing the Security of GitHub Copilot's Code Contributions. In *2022 IEEE Symposium on Security and Privacy (SP)*. 754–768. https://doi.org/10.1109/SP46214.2022.9833571

[15] Jiao Sun, Q. Vera Liao, Michael Muller, Mayank Agarwal, Stephanie Houde, Kartik Talamadupula, and Justin D. Weisz. 2022. Investigating Explainability of Generative AI for Code through Scenario-Based Design. In *27th International Conference on Intelligent User Interfaces* (Helsinki, Finland) *(IUI '22)*. Association for Computing Machinery, New York, NY, USA, 212–228. https://doi.org/10.1145/3490099.3511119

[16] Helena Vasconcelos, Matthew Jörke, Madeleine Grunde-McLaughlin, Ranjay Krishna, Tobias Gerstenberg, and Michael S. Bernstein. 2022. When Do XAI Methods Work? A Cost-Benefit Approach to Human-AI Collaboration. In *CHI Workshop on Trust and Reliance in AI-Human Teams* (New Orleans, USA). ACM. https://chi-trait.github.io/papers/CHI_TRAIT_2022_Paper_44.pdf

[17] Christopher D Wickens, Benjamin A Clegg, Alex Z Vieane, and Angelia L Sebok. 2015. Complacency and automation bias in the use of imperfect automation. *Human factors* 57, 5 (2015), 728–739.

[18] Albert Ziegler, Eirini Kalliamvakou, X. Alice Li, Andrew Rice, Devon Rifkin, Shawn Simister, Ganesh Sittampalam, and Edward Aftandilian. 2022. Productivity Assessment of Neural Code Completion. In *Proceedings of the 6th ACM SIGPLAN International Symposium on Machine Programming* (San Diego, CA, USA) *(MAPS 2022)*. Association for Computing Machinery, New York, NY, USA, 21–29. https://doi.org/10.1145/3520312.3534864