

What about ChatGPT/CoPilot/DeepSeek/LLMs in general?

- 3 arguments on why you should use them, but *after* you have passed this course (so NOT during this course)
- **Point 1**
Modes of using LLMs
 - *Exploration* mode
 - You don't know what to do, and you delegate this to the LLM → not effective
 - *Acceleration* mode
 - You know how to program and what to do, and you use the LLM to get there faster → very effective
 - Take home message: you need to learn how to program before benefitting from LLM!



Grounded Copilot: How Programmers Interact with Code-Generating Models

SHRADDHA BARKE*, UC San Diego, USA
MICHAEL B. JAMES*, UC San Diego, USA
NADIA POLIKARPOVA, UC San Diego, USA

Powered by recent advances in code-generating models, AI assistants like Github Copilot promise to change the face of programming forever. But what is this new face of programming? We present the first grounded theory analysis of how programmers interact with Copilot, based on observing 20 participants—with a range of prior experience using the assistant—as they solve diverse programming tasks across four languages. Our main finding is that interactions with programming assistants are *bimodal*: in *acceleration mode*, the programmer knows what to do next and uses Copilot to get there faster; in *exploration mode*, the programmer is unsure how to proceed and uses Copilot to explore their options. Based on our theory, we provide recommendations for improving the usability of future AI programming assistants.

CCS Concepts: • **Human-centered computing** → HCI theory, concepts and models; • **Software and its engineering** → Software creation and management.

Additional Key Words and Phrases: Program Synthesis, AI Assistants, Grounded Theory

ACM Reference Format:

Shraddha Barke, Michael B. James, and Nadia Polikarpova. 2023. Grounded Copilot: How Programmers Interact with Code-Generating Models. *Proc. ACM Program. Lang.* 7, OOPSLA1, Article 78 (April 2023), 27 pages. <https://doi.org/10.1145/3586030>

1 INTRODUCTION

The dream of an “AI assistant” working alongside the programmer has captured our imagination for several decades now, giving rise to a rich body of work from both the programming languages [Ferdowsifard et al. 2020; Miltner et al. 2019; Ni et al. 2021; Raychev et al. 2014] and the machine learning [Guo et al. 2021; Kalyan et al. 2018; Xu et al. 2020] communities. Thanks to recent breakthroughs in large language models (LLMs) [Li et al. 2022; Vaswani et al. 2017] this dream finally seems within reach. OpenAI’s Codex model [Chen et al. 2021], which contains 12 billion model parameters and is trained on 54 million software repositories on GitHub, is able to correctly solve 30–70% of novel Python problems, while DeepMind’s AlphaCode [Li et al. 2022] ranked in the top 54.3% among 5000 human programmers on the competitive programming platform Codeforces. With this impressive performance, large code-generating models are quickly escaping research labs to power industrial programming assistant tools, such as Github Copilot [Friedman 2021].

The growing adoption of these tools gives rise to questions about the nature of AI-assisted programming: *What kinds of tasks do programmers need assistance with? How do programmers prefer to communicate their intent to the tool? How do they validate the generated code to determine its*

*Equal contribution

Authors’ addresses: Shraddha Barke, UC San Diego, USA, sbarke@ucsd.edu; Michael B. James, UC San Diego, USA, m3james@ucsd.edu; Nadia Polikarpova, UC San Diego, USA, npolikarpova@ucsd.edu.



This work is licensed under a Creative Commons Attribution 4.0 International License.

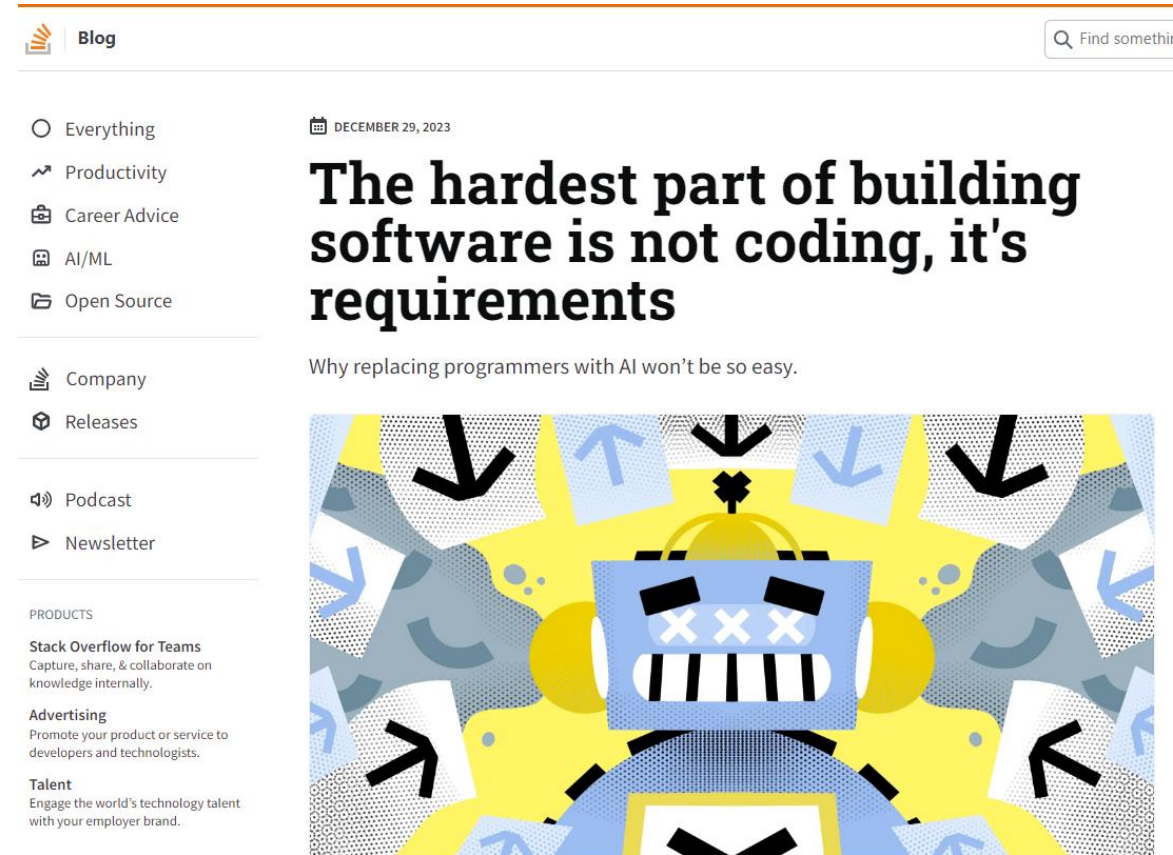
© 2023 Copyright held by the owner/author(s).

2475-1421/2023/4-ART78

<https://doi.org/10.1145/3586030>

What about ChatGPT/CoPilot/DeepSeek/LLMs in general? (cont.)

- **Point 2**
LLMs cannot really help you for the project/exam: this course is about teamwork, communication, quality, and reliability
- LLMs cannot offer any of this.
- <https://stackoverflow.blog/2023/12/29/the-hardest-part-of-building-software-is-not-coding-its-requirements/>



The screenshot shows a blog post on the Stack Overflow website. The page has a white background with a blue header. On the left, there is a navigation menu with icons and text for 'Blog', 'Everything', 'Productivity', 'Career Advice', 'AI/ML', 'Open Source', 'Company', 'Releases', 'Podcast', and 'Newsletter'. Below the menu, there are sections for 'PRODUCTS' including 'Stack Overflow for Teams', 'Advertising', and 'Talent'. The main content area on the right features a date 'DECEMBER 29, 2023' and the title 'The hardest part of building software is not coding, it's requirements'. Below the title is a sub-headline 'Why replacing programmers with AI won't be so easy.' and a large, colorful illustration of a person wearing a blue hard hat and a blue shirt with a white 'X' on the chest, surrounded by yellow and blue geometric shapes and arrows.

[Ed. note: While we take some time to rest up over the holidays and prepare for next year, we are re-publishing our top ten posts for the year. Please enjoy our favorite work this year and we'll see you in 2024.]

With all the articles about just how amazing all the developments in AI have been, there's plenty of hand wringing around the possibility that we, as software developers, could soon be out of a job, replaced by artificial intelligence. They imagine all the business execs and product researchers will bypass most or all of their software developers and asking AI directly to build exactly what they think

What about ChatGPT/CoPilot/DeepSeek/LLMs in general? (cont.)

- **Point 3**

Comprehension vs writing. LLMs spit out code that, unless understood in depth, should not be trusted. Understanding the implications of code can be much more difficult than writing the code yourself.

- <https://medium.com/bits-and-behavior/large-language-models-will-change-programming-a-little-81445778d957>
- <https://dl.acm.org/doi/10.1145/3442188.3445922>

ease. But that would be the hype talking. The reality is that while writing code is hard, the harder part for students (and really anyone) is understanding how it executes and then making decisions about what it should do differently. Program comprehension is what makes APIs hard to use (because they intentionally hide what they do, capturing behavior only through poorly written natural language). It's what makes programming



On the Dangers of Stochastic Parrots: Can Language Models Be Too Big? 🦜

Emily M. Bender*
ebender@uw.edu
University of Washington
Seattle, WA, USA

Angelina McMillan-Major
aymm@uw.edu
University of Washington
Seattle, WA, USA

Timnit Gebru*
timnit@blackinai.org
Black in AI
Palo Alto, CA, USA

Shmargaret Shmitchell
shmargaret.shmitchell@gmail.com
The Aether

ABSTRACT

The past 3 years of work in NLP have been characterized by the development and deployment of ever larger language models, especially for English. BERT, its variants, GPT-2/3, and others, most recently Switch-C, have pushed the boundaries of the possible both through architectural innovations and through sheer size. Using these pretrained models and the methodology of fine-tuning them for specific tasks, researchers have extended the state of the art on a wide array of tasks as measured by leaderboards on specific benchmarks for English. In this paper, we take a step back and ask: How big is too big? What are the possible risks associated with this technology and what paths are available for mitigating those risks? We provide recommendations including weighing the environmental and financial costs first, investing resources into curating and carefully documenting datasets rather than ingesting everything on the web, carrying out pre-development exercises evaluating how the planned approach fits into research and development goals and supports stakeholder values, and encouraging research directions beyond ever larger language models.

CCS CONCEPTS

• Computing methodologies → Natural language processing.

ACM Reference Format:

Emily M. Bender, Timnit Gebru, Angelina McMillan-Major, and Shmargaret Shmitchell. 2021. On the Dangers of Stochastic Parrots: Can Language Models Be Too Big? 🦜. In *Conference on Fairness, Accountability, and Transparency (FAccT '21)*, March 3–10, 2021, Virtual Event, Canada. ACM, New York, NY, USA, 14 pages. <https://doi.org/10.1145/3442188.3445922>

1 INTRODUCTION

One of the biggest trends in natural language processing (NLP) has been the increasing size of language models (LMs) as measured by the number of parameters and size of training data. Since 2018

*Joint first authors



This work is licensed under a Creative Commons Attribution International 4.0 License.
FAccT '21, March 3–10, 2021, Virtual Event, Canada
ACM ISBN 978-1-4503-8309-7/21/03.
<https://doi.org/10.1145/3442188.3445922>

alone, we have seen the emergence of BERT and its variants [39, 70, 74, 113, 146], GPT-2 [106], T-NLG [112], GPT-3 [25], and most recently Switch-C [43], with institutions seemingly competing to produce ever larger LMs. While investigating properties of LMs and how they change with size holds scientific interest, and large LMs have shown improvements on various tasks (§2), we ask whether enough thought has been put into the potential risks associated with developing them and strategies to mitigate these risks.

We first consider environmental risks. Echoing a line of recent work outlining the environmental and financial costs of deep learning systems [129], we encourage the research community to prioritize these impacts. One way this can be done is by reporting costs and evaluating works based on the amount of resources they consume [57]. As we outline in §3, increasing the environmental and financial costs of these models doubly punishes marginalized communities that are least likely to benefit from the progress achieved by large LMs and most likely to be harmed by negative environmental consequences of its resource consumption. At the scale we are discussing (outlined in §2), the first consideration should be the environmental cost.

Just as environmental impact scales with model size, so does the difficulty of understanding what is in the training data. In §4, we discuss how large datasets based on texts from the Internet overrepresent hegemonic viewpoints and encode biases potentially damaging to marginalized populations. In collecting ever larger datasets we risk incurring documentation debt. We recommend mitigating these risks by budgeting for curation and documentation at the start of a project and only creating datasets as large as can be sufficiently documented.

As argued by Bender and Koller [14], it is important to understand the limitations of LMs and put their success in context. This not only helps reduce hype which can mislead the public and researchers themselves regarding the capabilities of these LMs, but might encourage new research directions that do not necessarily depend on having larger LMs. As we discuss in §5, LMs are not performing natural language understanding (NLU), and only have success in tasks that can be approached by manipulating linguistic form [14]. Focusing on state-of-the-art results on leaderboards without encouraging deeper understanding of the mechanism by which they are achieved can cause misleading results as shown