/ Blog

Home / AI & ML / GitHub Copilot

Inside GitHub: Working with the LLMs behind GitHub Copilot

Developers behind GitHub Copilot discuss what it was like to work with OpenAI's large language model and how it informed the development of Copilot as we know it today.

Sara Verdi · **@saraverdi**

May 17, 2023

Updated February 7, 2024

10 minutes

Share:

take apart deep learning models to make sense of them and how they learn, but this was the first time that a model truly astonished me." Though the emergent behavior of the model was somewhat surprising, it was obviously powerful. Powerful enough, in fact, to lead to the creation of GitHub Copilot.

Due to the growing interest in LLMs and generative AI models, we decided to speak to the researchers and engineers at GitHub who helped build the early versions of GitHub Copilot and talk through what it was like to work with different LLMs from OpenAI, and how model improvements have helped evolve GitHub Copilot to where it is today—and beyond.

A brief history of GitHub Copilot

In June 2020, OpenAI released GPT-3, an LLM that sparked intrigue in developer communities and beyond. Over at GitHub, this got the wheels turning for a project our engineers had only talked about before: code generation.

"Every six months or so, someone would ask in our meetings, 'Should we think about general purpose code generation,' but the answer was always 'No, it's too difficult, the current models just can't do it,'" says Albert Ziegler, a principal machine learning engineer and member of the GitHub Next research and development team.

But GPT-3 changed all that—suddenly the model was good enough to begin considering how a code generation tool might work.

"OpenAI gave us the API to play around with," Ziegler says. "We assessed it by giving it coding-like tasks and evaluated it in two different forms."

For the first form of evaluation, the GitHub Next team crowdsourced self-contained problems to help test the model. "The reason we don't do this anymore is because the models just got too good," Ziegler laughs.

In the beginning, the model could solve about half of the problems it was posed with, but soon enough, it was solving upwards of 90 percent of the problems.

This original testing method sparked the first ideas for how to harness the power of this model, and they began to conceptualize an AI-powered chatbot for developers to ask

"The moment we did that and saw how well it worked, the whole static question-and-answer modality was forgotten," he says. "This new approach was interactive and it was useful in almost every situation."

And with that, the development of GitHub Copilot began.

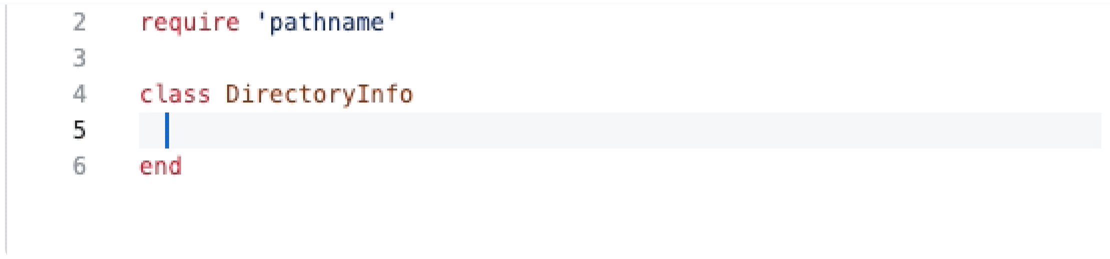
Exploring model improvements

To keep this project moving forward, GitHub returned to OpenAI to make sure that they could stay on track with the latest models. "The first model that OpenAI gave us was a Python-only model," Ziegler remembers. "Next we were delivered a JavaScript model and a multilingual model, and it turned out that the Javascript model had particular problems that the multilingual model did not. It actually came as a surprise to us that the multilingual model could perform so well. But each time, the models were just getting better and better, which was really exciting for GitHub Copilot's progress."

In 2021, OpenAI released the multilingual Codex model, which was built in partnership with GitHub. This model was an offshoot of GPT-3, so its original capability was generating natural language in response to text prompts. But what set the Codex model apart was that it was trained on billions of lines of public code—so that, in addition to natural language outputs, it also produced code suggestions.

This model was open for use via an API that businesses could build on, and while this breakthrough was huge for GitHub Copilot, the team needed to work on internal model improvements to ensure that it was as accurate as possible for end users.

As the GitHub Copilot product was prepared for launch as a technical preview, the team split off into further functional teams, and the Model Improvements team became responsible for monitoring and improving GitHub Copilot's quality through communicating with the underlying LLM. This team also set out to work on improving completion for users. Completion refers to when users accept and keep GitHub Copilot suggestions in their code, and there are several different levers that the Model Improvements team works on to increase completion, including prompt crafting and fine tuning.



```
require 'pathname'

class DirectoryInfo
  
end
```

An example of completion in action with GitHub Copilot.

Prompt crafting

When working with LLMs, you have to be very specific and intentional with your inputs to receive your desired output, and prompt crafting explores the art behind communicating these requests to get the optimal completion from the model.

“In very simple terms, the LLM is, at its core, just a document completion model. For training it was given partial documents and it learned how to complete them one token at a time. Therefore, the art of prompt crafting is really all about creating a ‘pseudo-document’ that will lead the model to a completion that benefits the customer,” John Berryman, a senior researcher of machine learning on the Model Improvements team explains. Since LLMs are trained on partial document completion, then if the partial document is code, then this completion capability lends itself well to code completion, which is, in its base form, exactly what GitHub Copilot does.

To better understand how the model could be applied to code completion, the team would provide the model with a file and evaluate the code completions it returned.

“Sometimes the results are ok, sometimes they are quite good, and sometimes the results seem almost magical,” Berryman says. “The secret is that we don’t just have to provide the model with the original file that the GitHub Copilot user is currently editing; instead we look for additional pieces of context inside the IDE that can hint the model towards better completions.”

He continues, “There have been several changes that helped get GitHub Copilot where it is today, but one of my favorite tricks was when we pulled similar texts in from the user’s neighboring editor tabs. That was a huge lift in our acceptance rate and characters retained.”

"The idea here is to make sure that we make developers more productive, but the way we do that is where things start to get interesting: we can make the user more productive by incorporating the way they think about code into the algorithm itself," Berryman says. "Where the developer might flip back and forth between tabs to reference code, we just can do that for them, and the completion is exactly what it would be if the user had taken all of the time to look that information up."

Fine-tuning

Fine-tuning is a technique used in AI to adapt and improve a pre-trained model for a specific task or domain. The process involves taking a pre-trained model that has been trained on a large dataset and training it on a smaller, more specific dataset that is relevant to a particular use case. This enables the model to learn and adapt to the nuances of the new data, thus improving its performance on the specific task.

These larger, more sophisticated LLMs can sometimes produce outputs that aren't necessarily helpful because it's hard to statistically define what constitutes a "good" response. It's also incredibly difficult to train a model like Codex that contains upwards of 170 billion parameters.

"Basically, we're training the underlying Codex model on a user's specific codebase to provide more focused, customized completions," Goudarzi adds.

"Our greatest challenge right now is to consider why the user rejects or accepts a suggestion," Goudarzi adds. "We have to consider what context, or information, that we served to the model caused the model to output something that was either helpful or not helpful. There's no way for us to really troubleshoot in the typical engineering way, but what we can do is figure out how to ask the right questions to get the output we desire."

Read more about how GitHub Copilot is getting better at understanding your code to provide a more customized coding experience here.

of those LLMs in house—GitHub Copilot has improved and gained new capabilities with chat functionality, voice-assisted development, and more via GitHub Copilot X on the horizon.

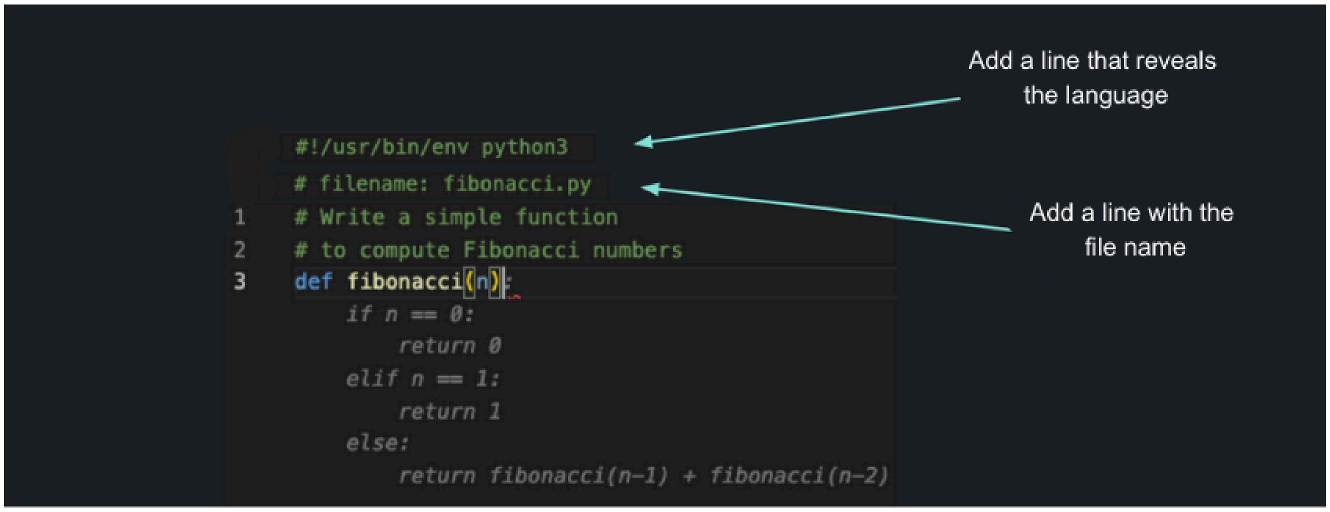
Johan Rosenkilde, a staff researcher on the GitHub Next team remembers, “When we received the latest model drops from OpenAI in the past, the improvements were good, but they couldn’t really be felt by the end user. When the third iteration of Codex dropped, you could *feel* it, especially when you were working with programming languages that are not one of the top five languages,” Rosenkilde says.

He continues, “I happened to be working on a programming competition with some friends on the weekend that model version was released, and we were programming with F#. In the first 24 hours, we evidently had the old model for GitHub Copilot, but then BOOM! Magic happened,” he laughs. “There was an incredibly noticeable difference.”

In the beginning, GitHub Copilot also had the tendency to suggest lines of code in a completely different programming language, which created a poor developer experience (for somewhat obvious reasons).

“You could be working in a C# project, then all of the sudden at the top of a new file, it would suggest Python code,” Rosenkilde explains. So, the team added a headline to the prompt which listed the language you were working in. “Now this had no impact when you were deep down in the file because Copilot could understand which language you were in. But at the top of the file, there could be some ambiguity, and those early models just defaulted to the top popular languages.”

About a month following that improvement, the team discovered that it was much more powerful to put the path of the file at the top of the document.


Add a line that reveals the language
Add a line with the file name

```
#!/usr/bin/env python3
# filename: fibonacci.py
# Write a simple function
# to compute Fibonacci numbers
def fibonacci(n):
    if n == 0:
        return 0
    elif n == 1:
        return 1
    else:
        return fibonacci(n-1) + fibonacci(n-2)
```

A diagram of the file path improvement.

"The end of the file name would give away the language in most cases, and in fact the file name could provide crucial, additional information," Rosenkilde says. "For example, the file might be named 'connectiondatabase.py.' Well that file is most likely about databases or connections, so you might want to import an SQL library, and that file was written in Python. So, that not only solved the language problem, but it also improved the quality and user experience by a surprising margin because GitHub Copilot could now suggest boilerplate code."

After a few more months of work, and several iterations, the team was able to create a component that lifted code from other files, which is a capability that had been talked about since the genesis of GitHub Copilot. Rosenkilde recalls, "this never really amounted to anything more than conversations or a draft pull request because it was so abstract. But then, Albert Ziegler built this component that looked at other files you have open in the IDE at that moment in time and scanned through those files for similar text to what's in your current cursor. This was a huge boost in code acceptance because suddenly, GitHub Copilot knew about other files."

What's next for GitHub Copilot

2023, GitHub announced the future of Copilot, GitHub Copilot X, our vision for an AI-powered developer experience. GitHub Copilot X aims to bring AI beyond the IDE to more components of the overall platform, such as docs and pull requests. LLMs are changing the ways that we interact with technology and how we work, and ideas like GitHub Copilot X are just an example of what these models, along with some dedicated training techniques, are capable of.

GitHub Copilot trial
Don't fly solo. Try for 30 days free.
Learn more ›

Tags: AI | AI Insights | generative AI | GitHub Copilot | How GitHub builds GitHub | LLM

Written by

Sara Verdi
@saraverdi

More on AI

Celebrating the GitHub Awards 2024 recipients 🎉

/ Blog

Laura Lindeman & Lee Reilly

Bringing developer choice to Copilot with Anthropic's Claude 3.5 Sonnet, Google's Gemini 1.5 Pro, and OpenAI's o1-preview

At GitHub Universe, we announced Anthropic's Claude 3.5 Sonnet, Google's Gemini 1.5 Pro, and OpenAI's o1-preview and o1-mini are coming to GitHub Copilot—bringing a new level of choice to every developer.

Thomas Dohmke

Related posts

work and changing how they spend their days.

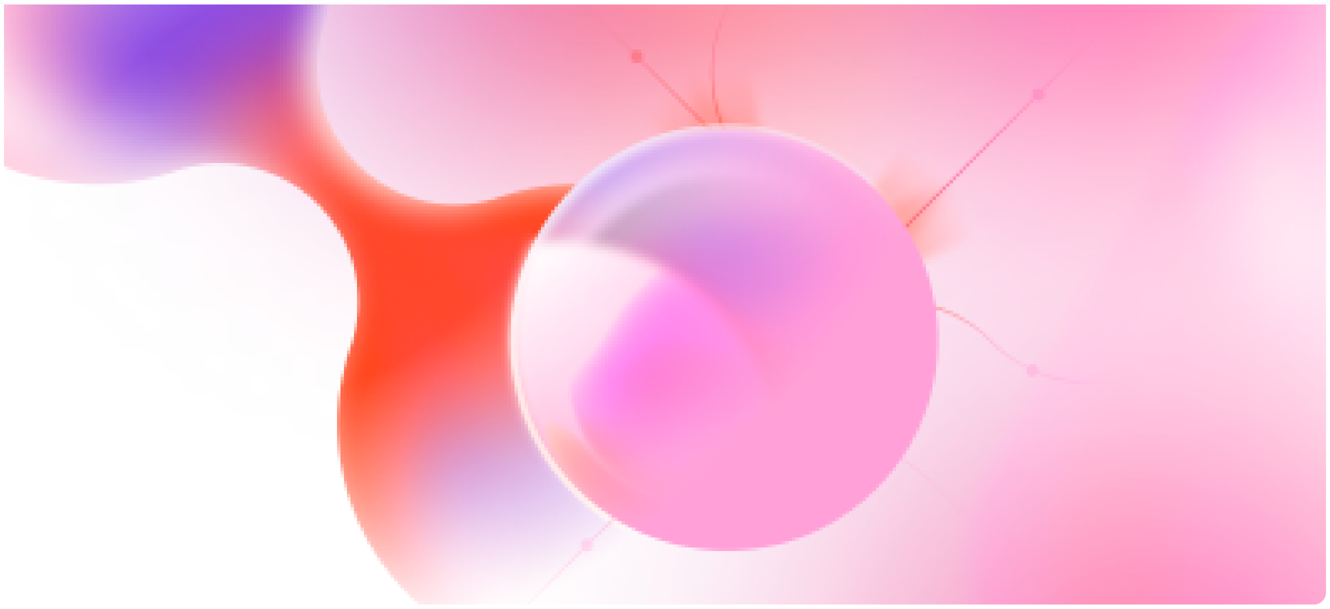
Klint Finley

5 tips and tricks when using GitHub Copilot Workspace

GitHub Next launched the technical preview for GitHub Copilot Workspace in April 2024. Since then, we've been listening to the community, learning, and have some tips to share on how to get the most out of it!

Chris Reddington & Cole Bemis

/ Blog



How students teamed up to decode 2,000-year-old texts using AI

Students used GitHub Copilot to decode ancient texts buried in Mount Vesuvius, achieving a groundbreaking historical breakthrough. This is their journey, the technology behind it, and the power of collaboration.

Juan Pablo Flores Cortés

Explore more from GitHub

Docs

Everything you need to master GitHub, all in one place.

Go to Docs

/ Blog

GitHub

Build what’s next on GitHub, the place for anyone from anywhere to build anything.

Start building >

Customer stories

Meet the companies and engineering teams that build with GitHub.

Learn more

GitHub Universe 2024

Get tickets to the 10th anniversary of our global developer event on AI, DevEx, and security.

Get tickets

/ Blog

We do newsletters, too

Discover tips, technical guides, and best practices in our biweekly newsletter just for devs.

Your email address

■ Yes please, I'd like GitHub and affiliates to use my information for personalized communications, targeted advertising and campaign effectiveness. See the GitHub Privacy Statement for more details.

Product	Platform	Support	Company
Features	Developer API	Docs	About
Security	Partners	Community Forum	Blog
Enterprise	Atom	Training	Careers
Customer Stories	Electron	Status	Press
Pricing	GitHub Desktop	Contact	Shop
Resources			

/ Blog