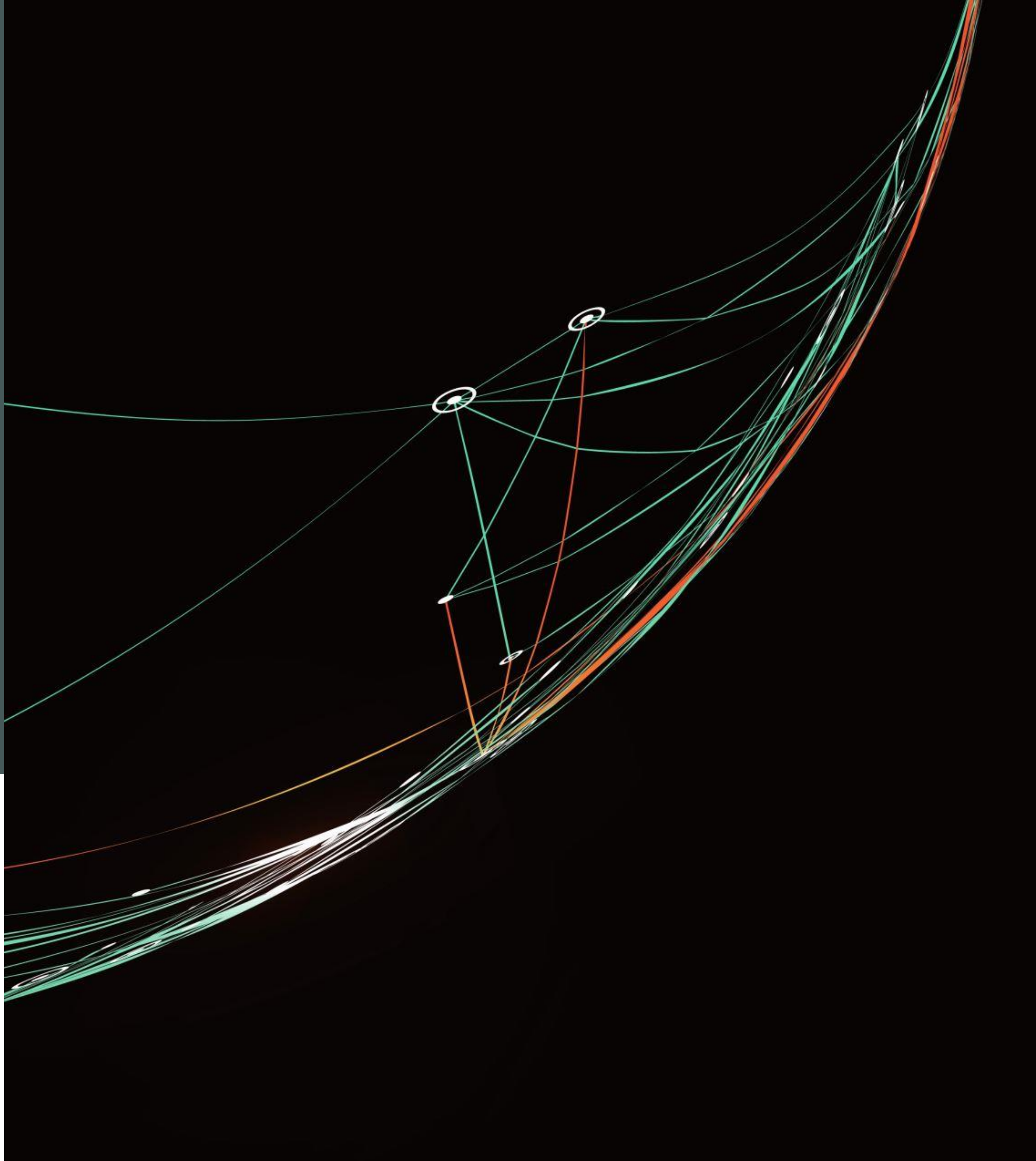
Beyond Code Generation

Towards Next-Generation AI for SE

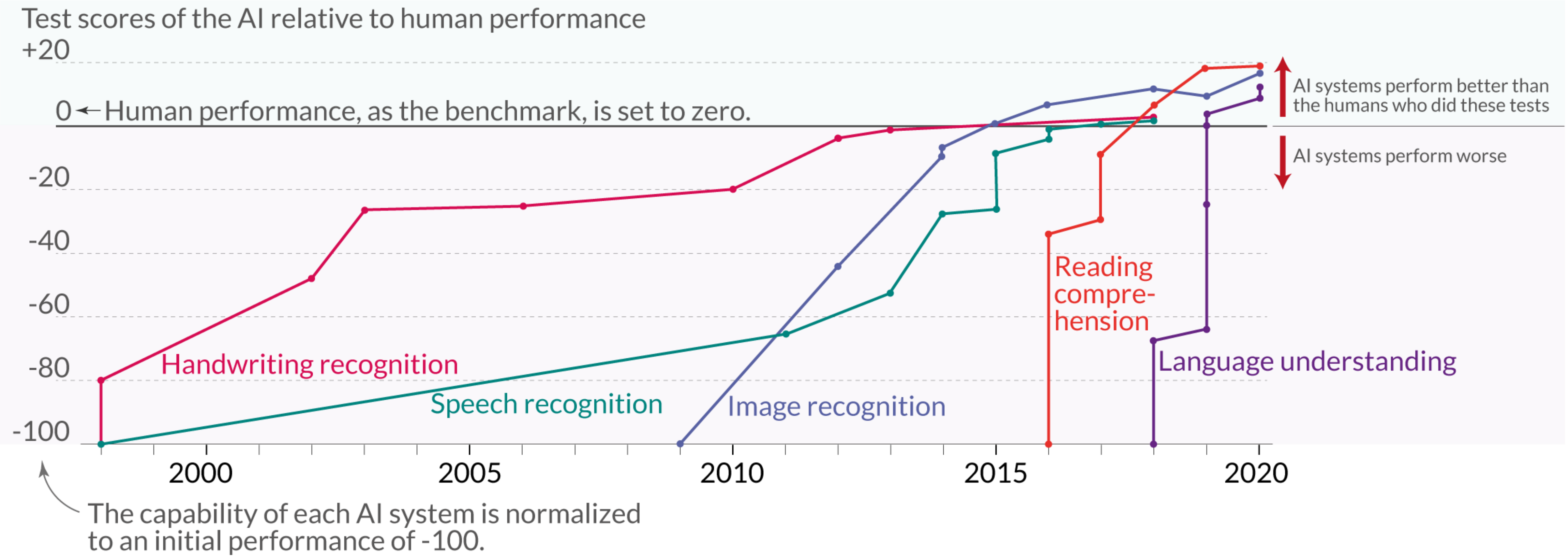
Vincent J. Hellendoorn

December 3rd, 2023

MAPS @ FSE

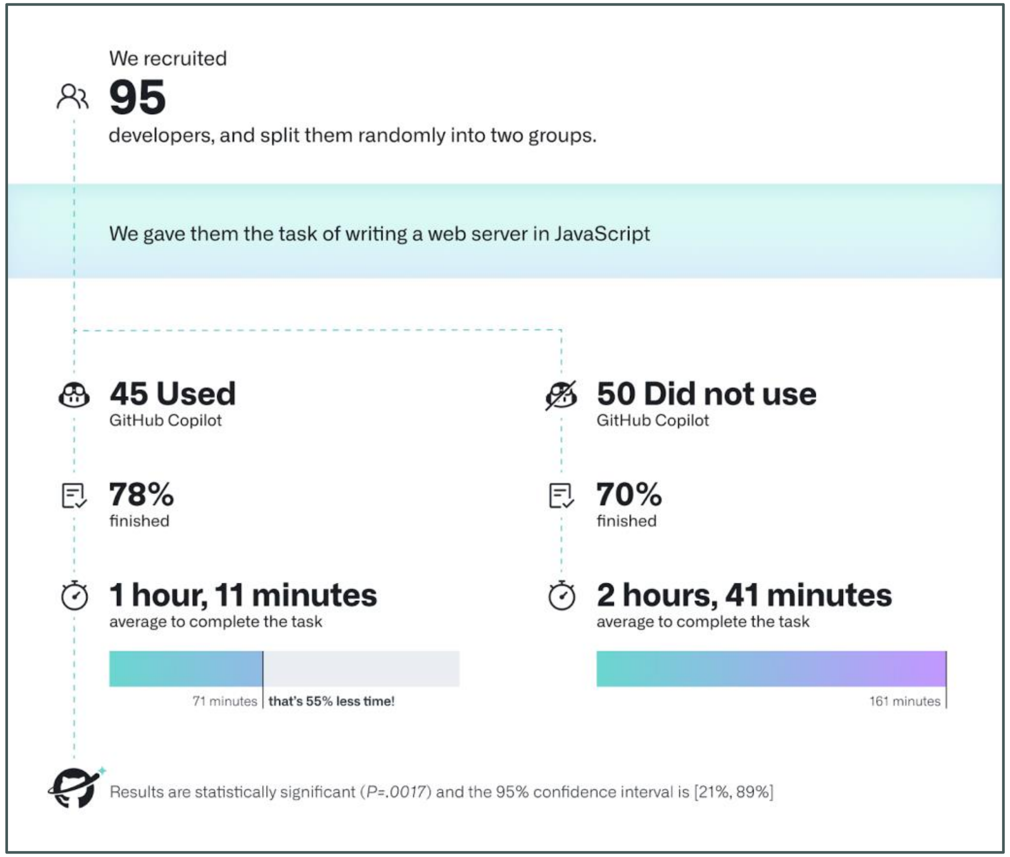


Where We Were



Max Roser (2022) - "The brief history of artificial intelligence: The world has changed fast – what might be next?" Published online at OurWorldInData.org. Retrieved from: 'https://ourworldindata.org/brief-history-of-ai' [Online Resource]

Where We Were



We recruited

95

developers, and split them randomly into two groups.

We gave them the task of writing a web server in JavaScript

45 Used
GitHub Copilot

78%
finished

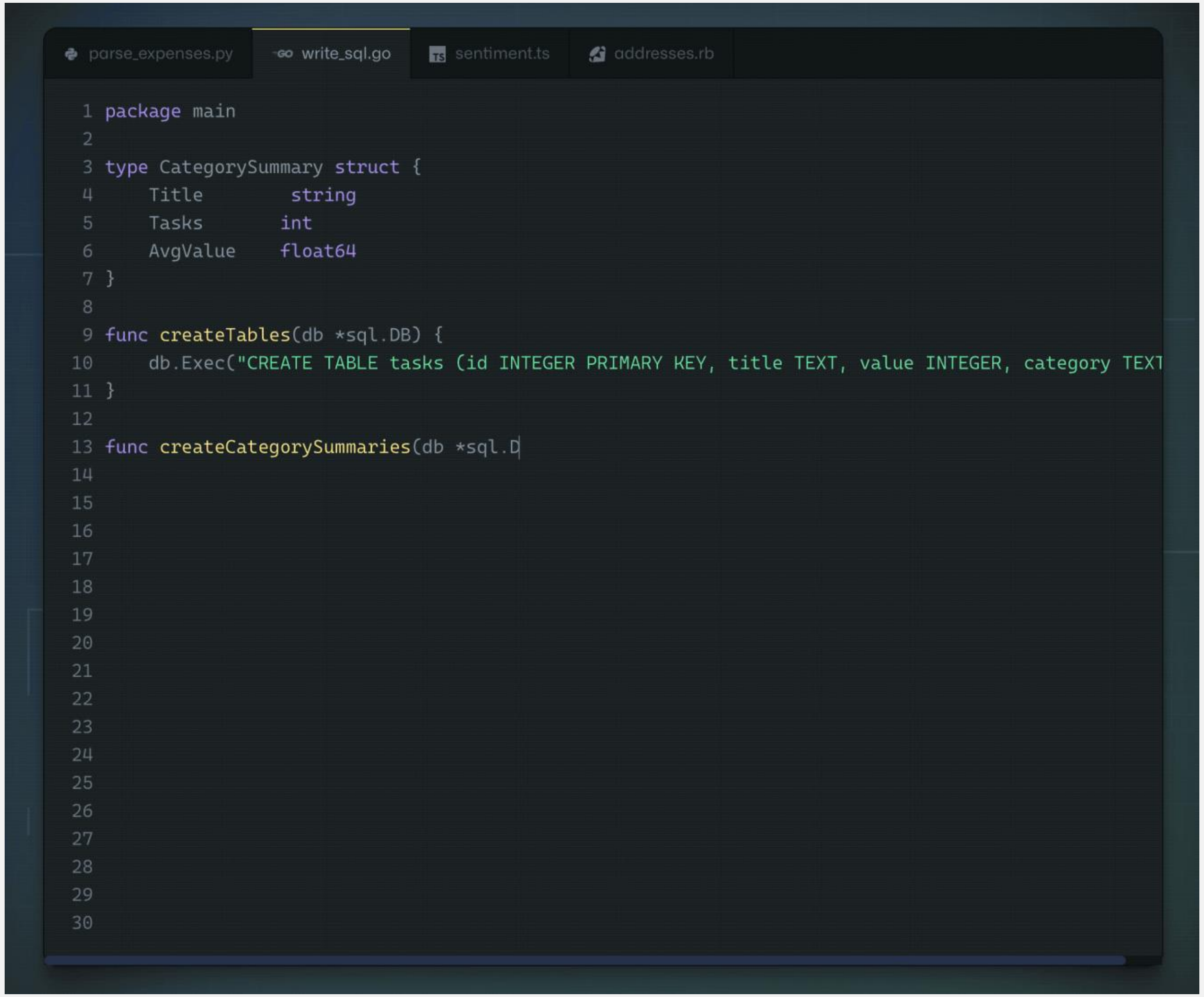
1 hour, 11 minutes
average to complete the task

50 Did not use
GitHub Copilot

70%
finished

2 hours, 41 minutes
average to complete the task

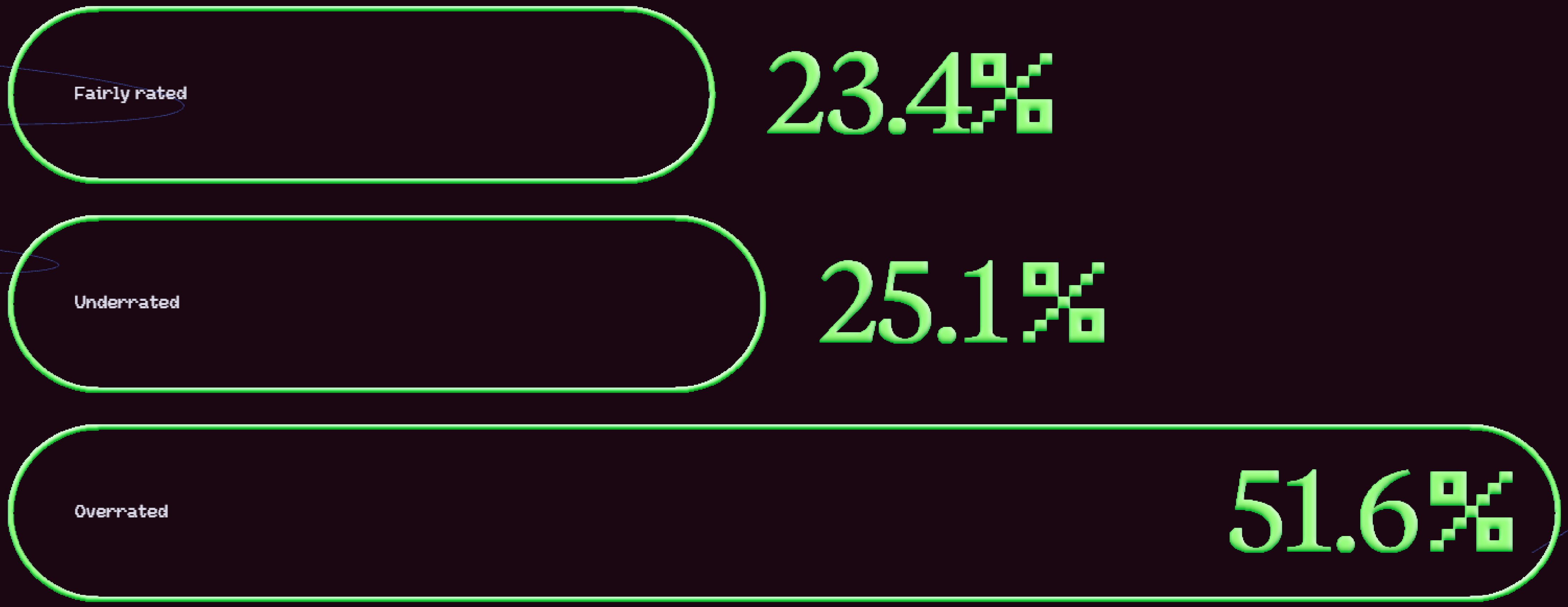
Results are statistically significant (*P=.0017*) and the 95% confidence interval is [21%, 89%]



```go
package main

type CategorySummary struct {
    Title        string
    Tasks       int
    AvgValue    float64
}

func createTables(db *sql.DB) {
    db.Exec("CREATE TABLE tasks (id INTEGER PRIMARY KEY, title TEXT, value INTEGER, category TEXT
}

func createCategorySummaries(db *sql.D
```

https://copilot.github.com/ https://github.blog/2022-09-07-research-quantifying-github-copilots-impact-on-developer-productivity-and-happiness/

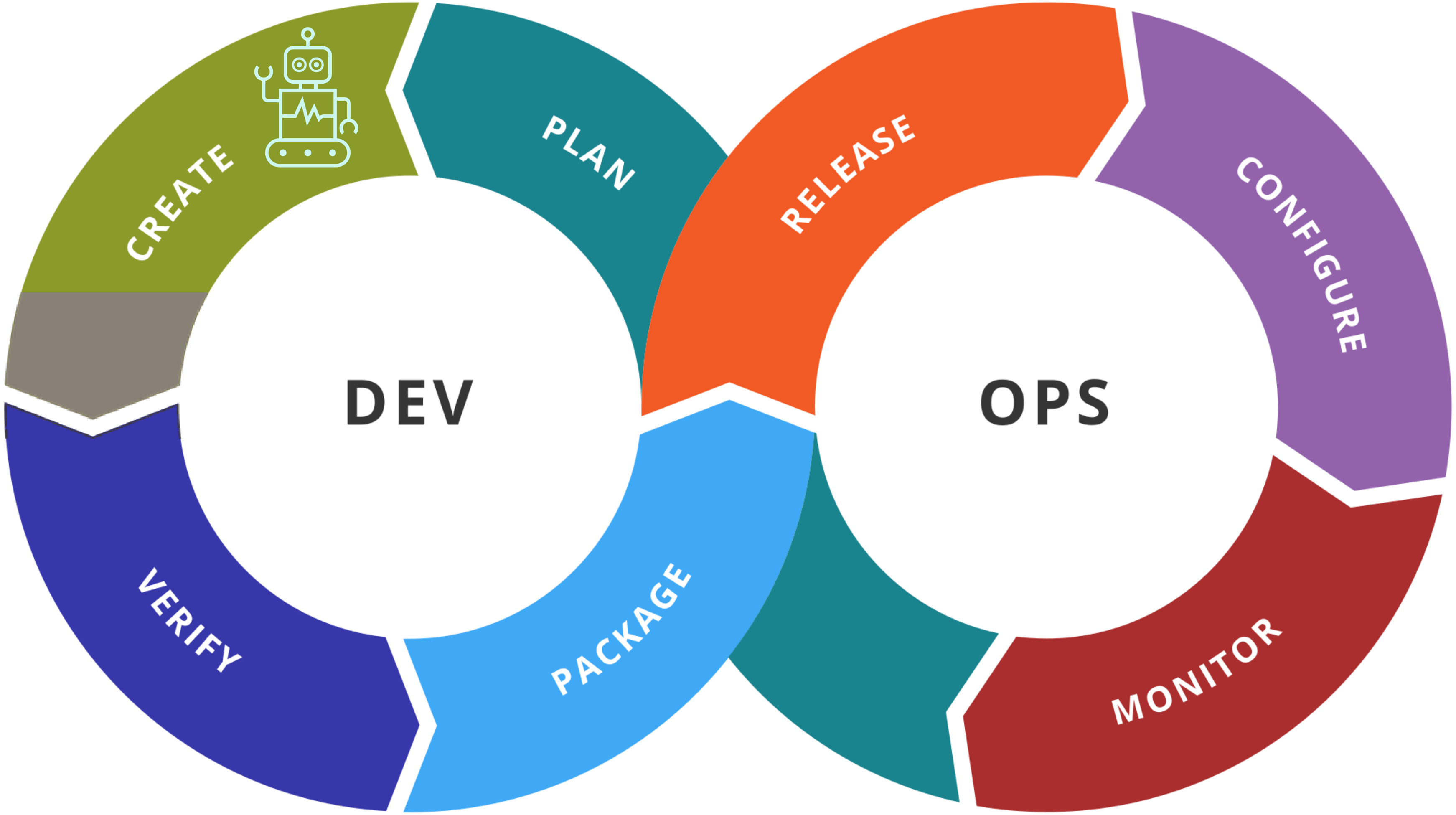
Where We Aren't



⟍ How fairly rated is AI?

[data viz 1.1] / % by category

Today:
Where We Are, and Are Going



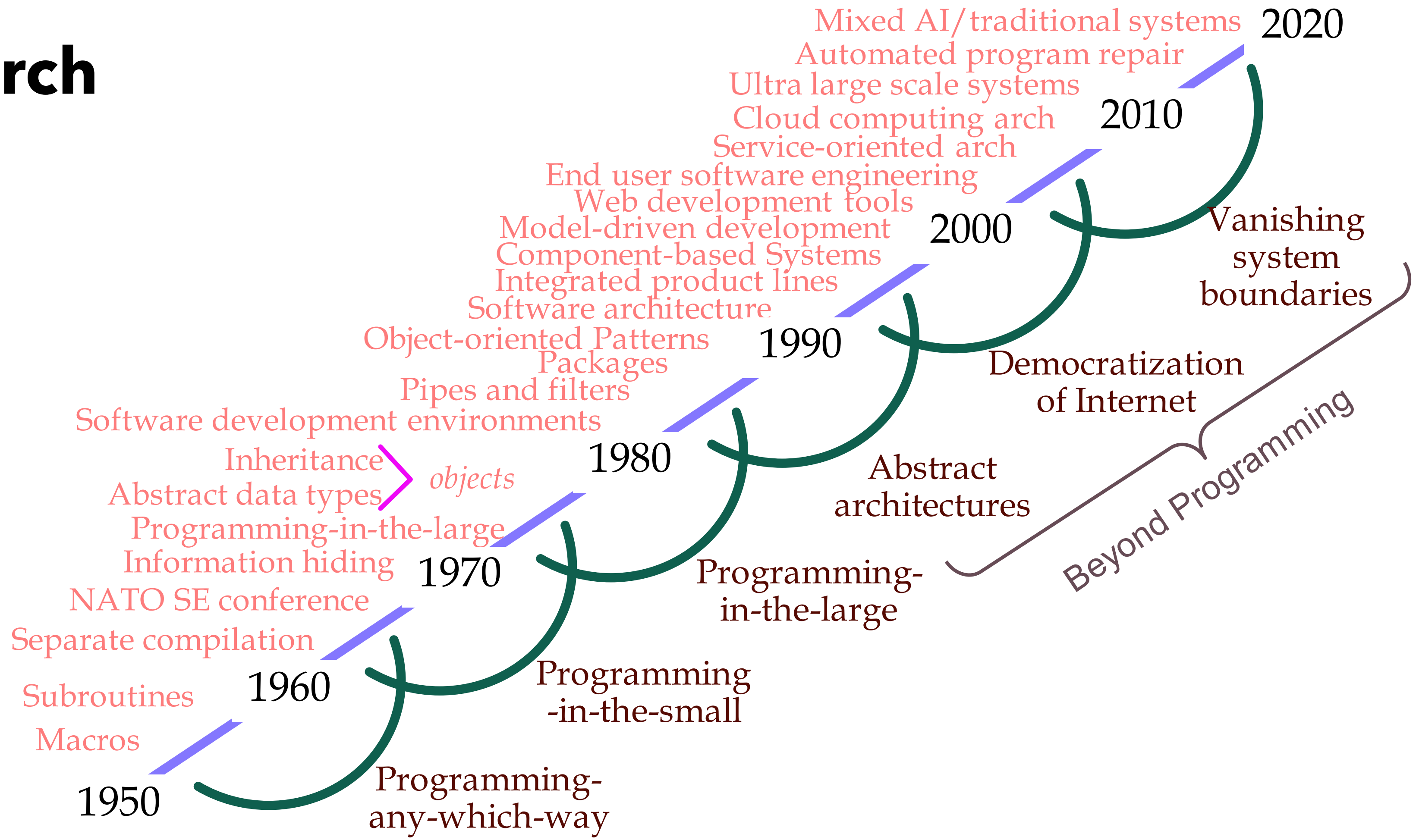
https://en.wikipedia.org/wiki/DevOps_toolchain#/media/File:Devops-toolchain.svg

It’s time to move beyond writing assistants

1. Start by reflecting on the olden days (<2020)
2. Then, discuss how LLMs changed the picture
3. Next, highlight recent progress & trends
4. Finally, promises, challenges, needs & tips

Software Development Always Changes

And so does **SE research**



Mixed AI/traditional systems
2020
Automated program repair
Ultra large scale systems
2010
Cloud computing arch
Service-oriented arch
End user software engineering
Web development tools
Model-driven development
2000
Vanishing system boundaries
Component-based Systems
Integrated product lines
Software architecture
Object-oriented Patterns
1990
Packages
Democratization of Internet
Pipes and filters
Software development environments
Inheritance
Abstract data types
objects
1980
Abstract architectures
Beyond Programming
Programming-in-the-large
Information hiding
1970
Programming-in-the-large
NATO SE conference
Separate compilation
1960
Programming -in-the-small
Subroutines
Macros
1950
Programming-any-which-way

“First Wave” of ML for SE

Hand-extracted features fed to **off-the-shelf learners**

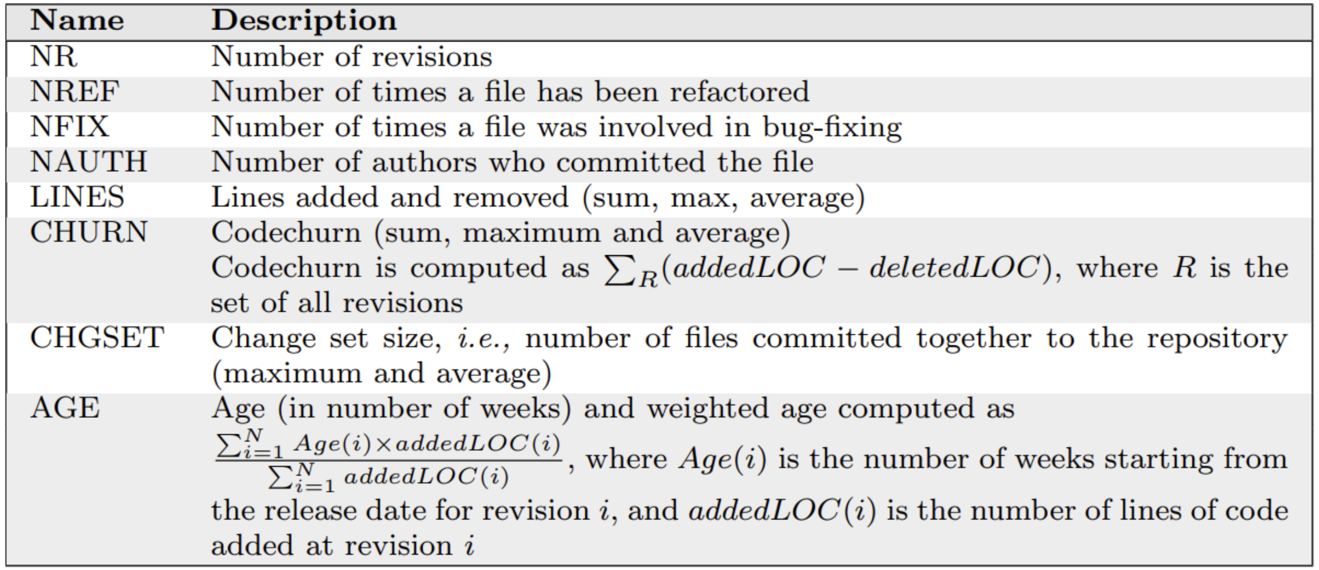
Multiple, generic models (e.g., decision trees)

Practitioner focuses on **features & statistics**

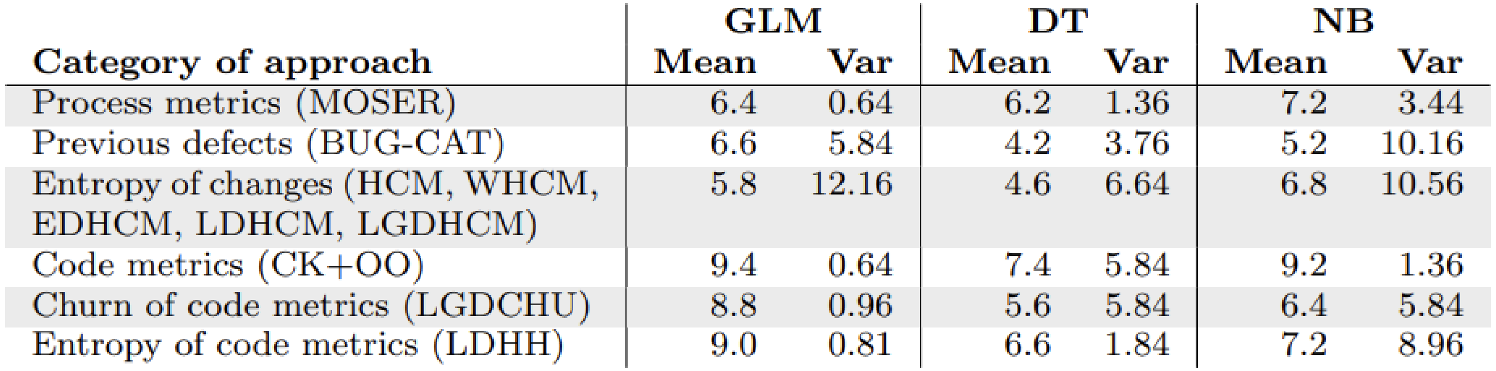
Following the taxonomy from Allamanis et al., https://arxiv.org/abs/1709.06182
Image by Stable Diffusion XL

“First Wave” of ML for SE

What that looked like:



Name	Description
NR	Number of revisions
NREF	Number of times a file has been refactored
NFIX	Number of times a file was involved in bug-fixing
NAUTH	Number of authors who committed the file
LINES	Lines added and removed (sum, max, average)
CHURN	Codechurn (sum, maximum and average) Codechurn is computed as $\sum_R (addedLOC - deletedLOC)$, where R is the set of all revisions
CHGSET	Change set size, *i.e.*, number of files committed together to the repository (maximum and average)
AGE	Age (in number of weeks) and weighted age computed as $\frac{\sum_{i=1}^{N} Age(i) \times addedLOC(i)}{\sum_{i=1}^{N} addedLOC(i)}$, where $Age(i)$ is the number of weeks starting from the release date for revision i, and $addedLOC(i)$ is the number of lines of code added at revision i



	GLM		DT		NB	
Category of approach	**Mean**	**Var**	**Mean**	**Var**	**Mean**	**Var**
Process metrics (MOSER)	6.4	0.64	6.2	1.36	7.2	3.44
Previous defects (BUG-CAT)	6.6	5.84	4.2	3.76	5.2	10.16
Entropy of changes (HCM, WHCM, EDHCM, LDHCM, LGDHCM)	5.8	12.16	4.6	6.64	6.8	10.56
Code metrics (CK+OO)	9.4	0.64	7.4	5.84	9.2	1.36
Churn of code metrics (LGDCHU)	8.8	0.96	5.6	5.84	6.4	5.84
Entropy of code metrics (LDHH)	9.0	0.81	6.6	1.84	7.2	8.96

D’ambros, ESE, Evaluating Defect Prediction Approaches: A Benchmark and an Extensive Comparison

“First Wave” of ML for SE

Hand-extracted features fed to **off-the-shelf learners**

Pros:

- Useful for **almost any task** where decisions are made

Cons:

- Feature selection **limits performance**, requires manual effort
- Largely **inapplicable to code**

“Second Wave” of ML for SE

Learning **from** and for **Source Code**

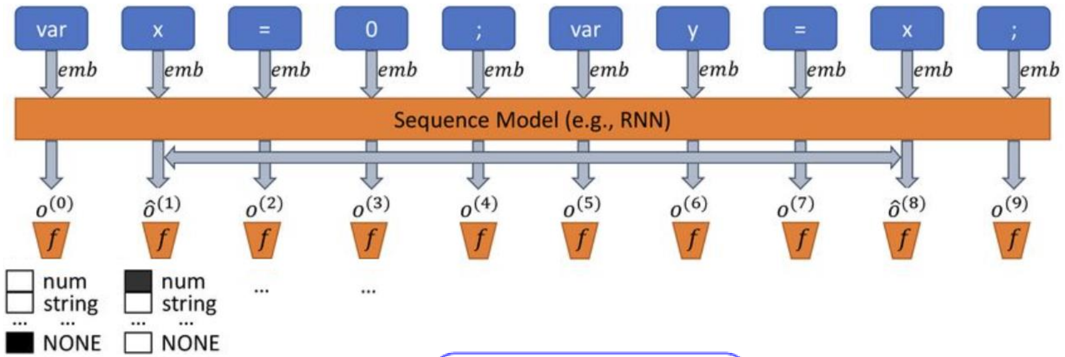
Feature learning is left to the models

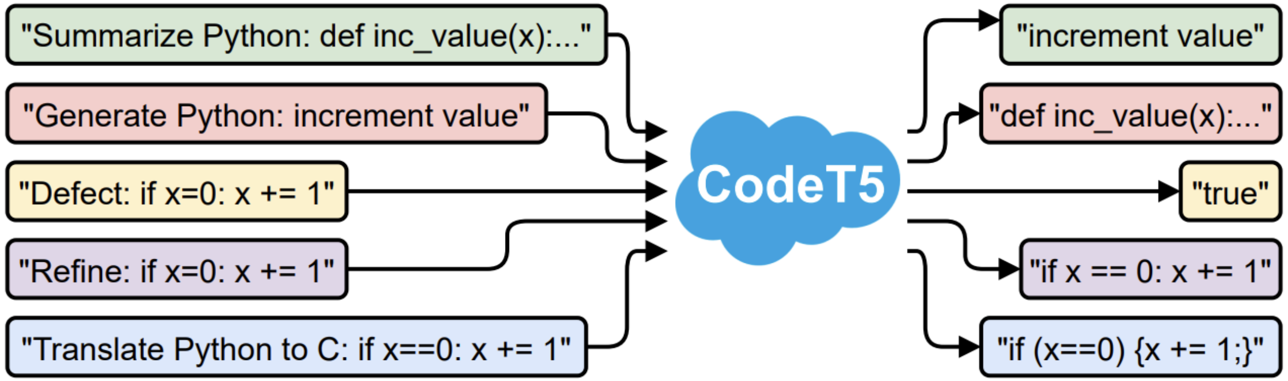
Model design is often **inspired by tasks**

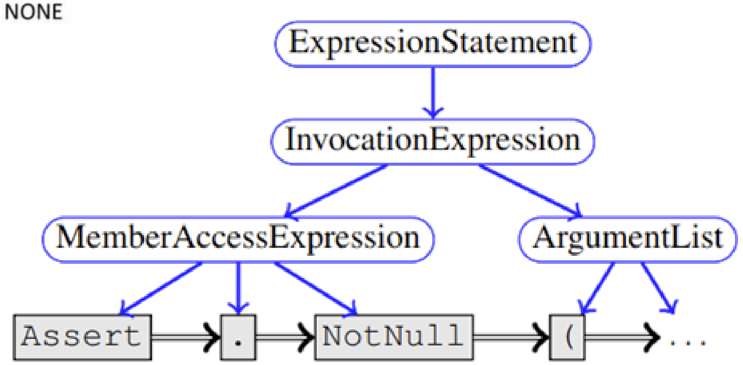
Image by Stable Diffusion XL

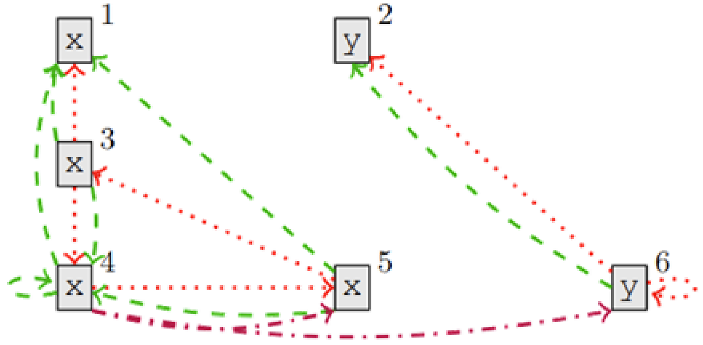
“Second Wave” of ML for SE

What that looked like:









Marc Brockschmidth, MSR. (Deep) Learning from Programs. Slides from: https://slideplayer.com/slide/16532816/
Allamanis et al., ICLR’18. Learning to Represent Programs with Graphs. https://arxiv.org/pdf/1711.00740.pdf
Wang et al., EMNLP’21. CodeT5: Identifier-aware Unified Pre-trained Encoder-Decoder Models for Code Understanding and Generation

“Second Wave” of ML for SE

Learning **from** and for **Source Code**

Pros:

- Increased **expressivity**, less manual effort

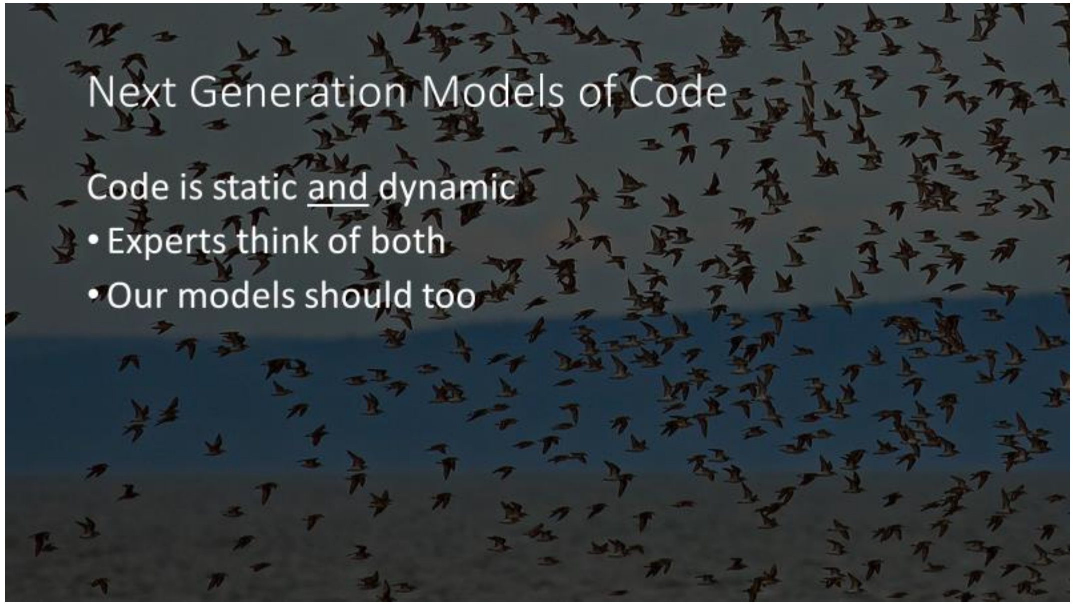
Cons:

- Learning limited by **dataset size**
- Limited **practical utility**

Image by Stable Diffusion XL

Obviously, the "third wave" is learning program semantics, right?

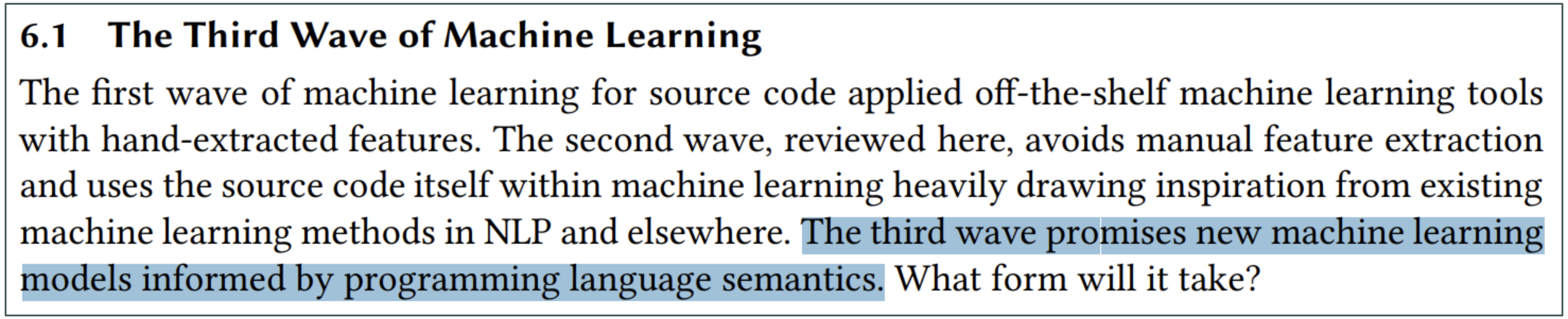
Right?



Next Generation Models of Code

Code is static and dynamic

- Experts think of both
- Our models should too

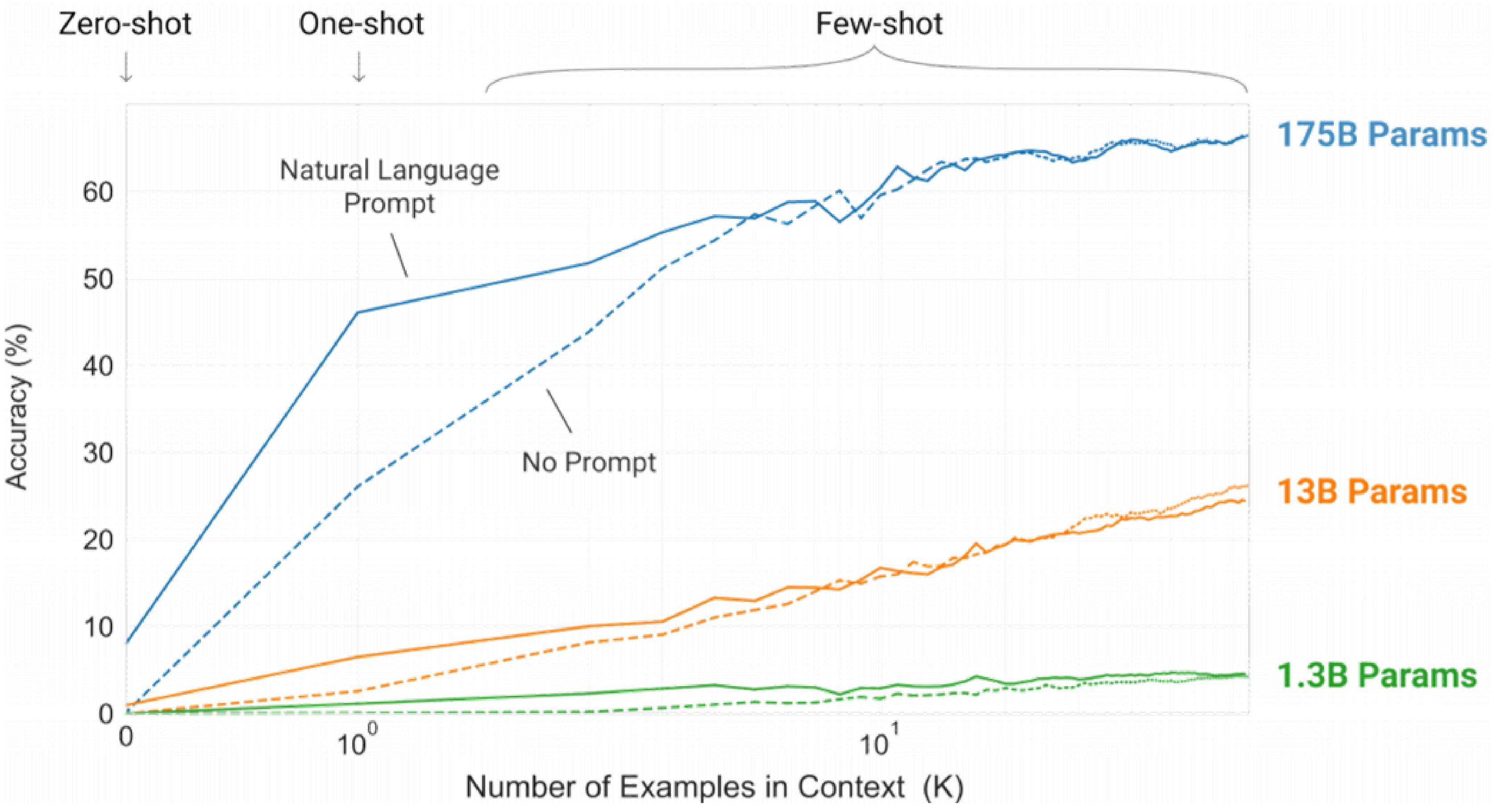


6.1 The Third Wave of Machine Learning

The first wave of machine learning for source code applied off-the-shelf machine learning tools with hand-extracted features. The second wave, reviewed here, avoids manual feature extraction and uses the source code itself within machine learning heavily drawing inspiration from existing machine learning methods in NLP and elsewhere. The third wave promises new machine learning models informed by programming language semantics. What form will it take?

Allamanis et al., https://arxiv.org/abs/1709.06182, slide from my job talk

The Bitter Lesson Strikes Again



GPT-3 (Brown et al., 2020), https://arxiv.org/pdf/2005.14165.pdf

“Third Wave” of ML for SE

Pretraining at **immense scale**

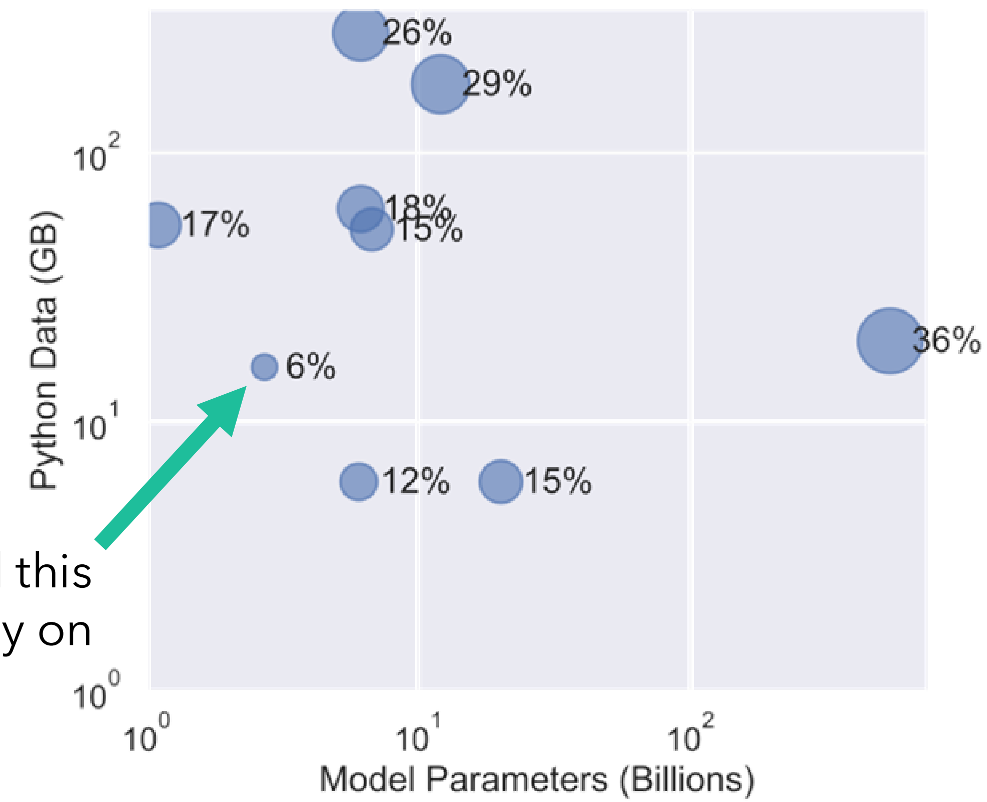
Not just on code

“Third Wave” of ML for SE

Pretraining at **immense scale**

Not just on code

In the world of **Large Language Models**, the goal is generation and the currency is compute



We trained this
one early on

https://dpfried.github.io/talks/programming-communication.pdff

“Third Wave” of ML for SE

Pretraining at **immense scale**

Pros:

- Can generate **large volumes of code** & text
- Extraordinary **representational power**

Cons:

- Inherently **generative**
- Extremely **data, compute hungry**

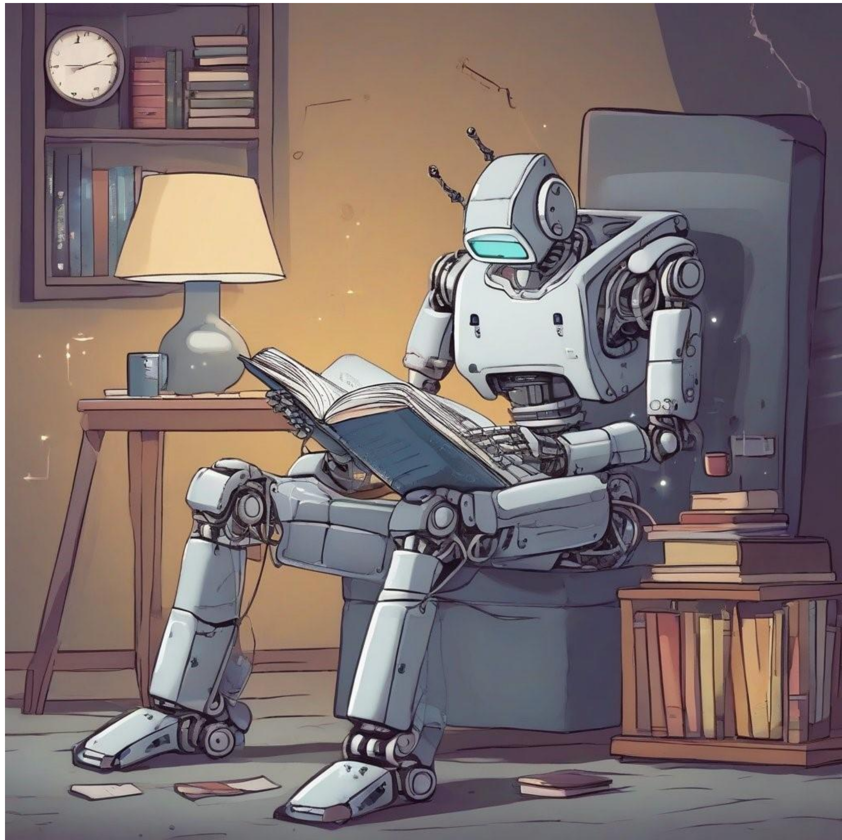
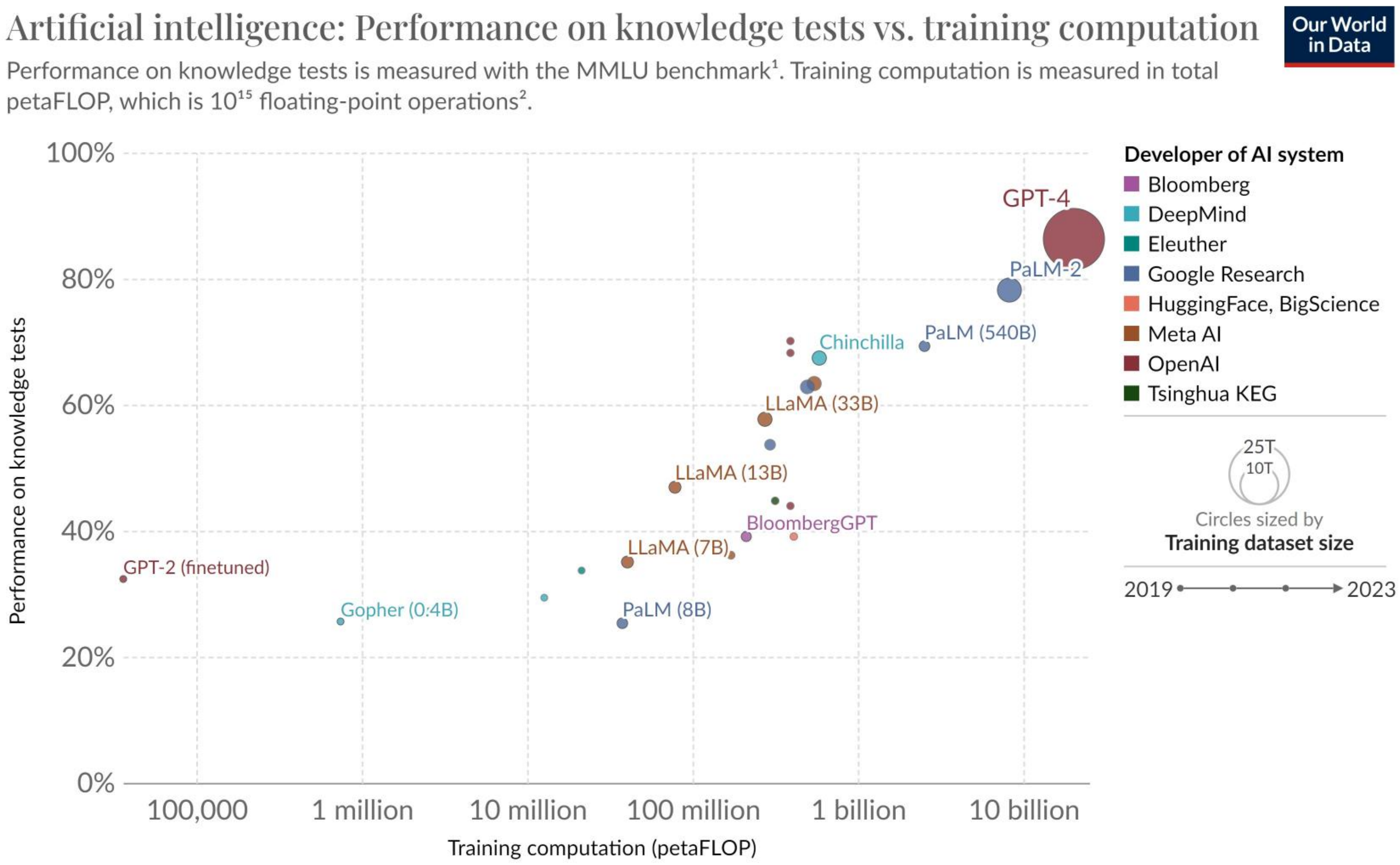


Image by Stable Diffusion XL

Well, We Kept Feeding It



Artificial intelligence: Performance on knowledge tests vs. training computation

Performance on knowledge tests is measured with the MMLU benchmark[1]. Training computation is measured in total petaFLOP, which is 10^{15} floating-point operations[2].

Published online at OurWorldInData.org. Retrieved from: 'https://ourworldindata.org/artificial-intelligence' [Online Resource]

Where We Are: an Inflection Point

We used to model **any process**, but the models were bad

Now we can model **one process insanely well**

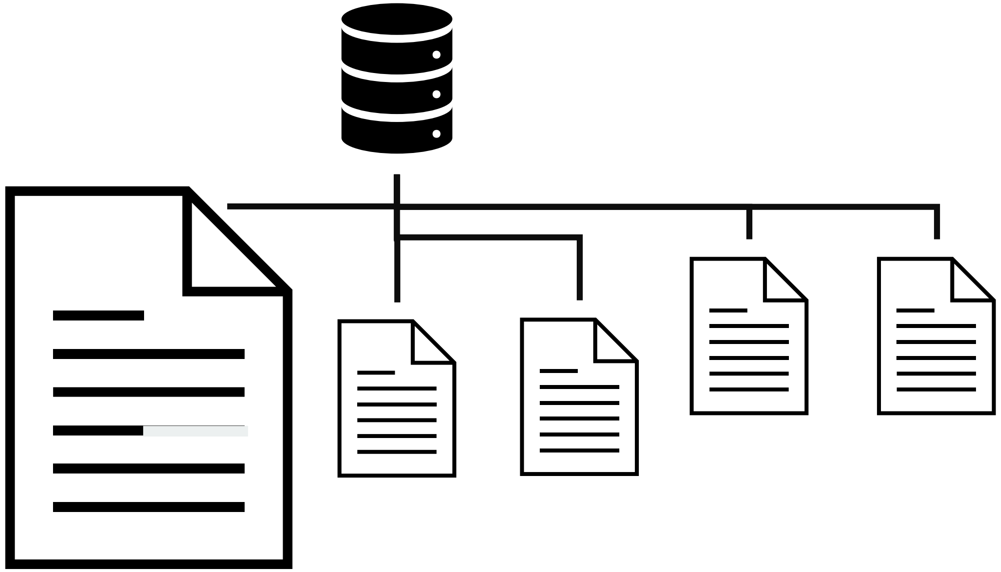
Let's spread the love

Published online at OurWorldInData.org. Retrieved from: 'https://ourworldinda

Five Challenges for “AI for Code”

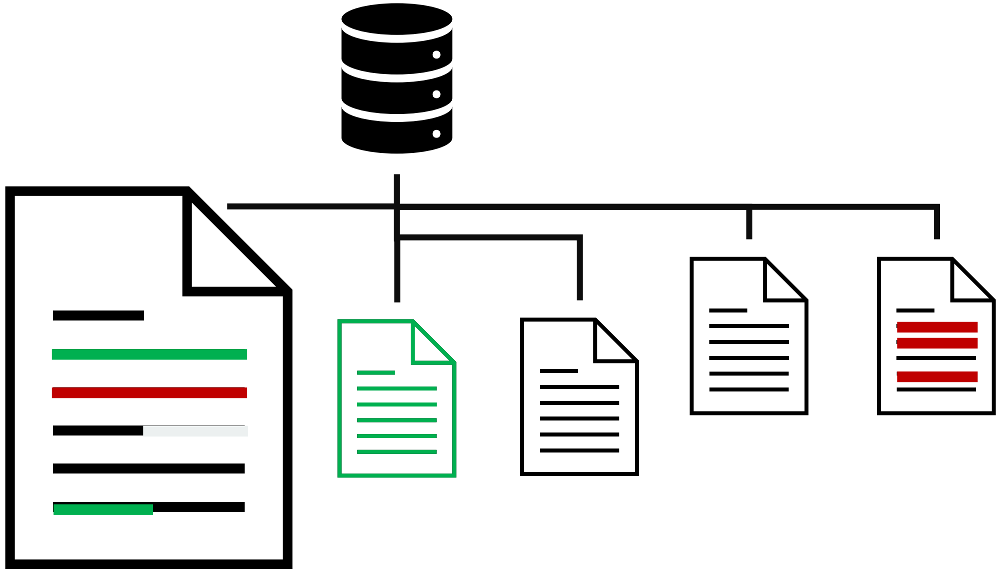
Five Challenges for “AI for Code”

1. Expanding Context



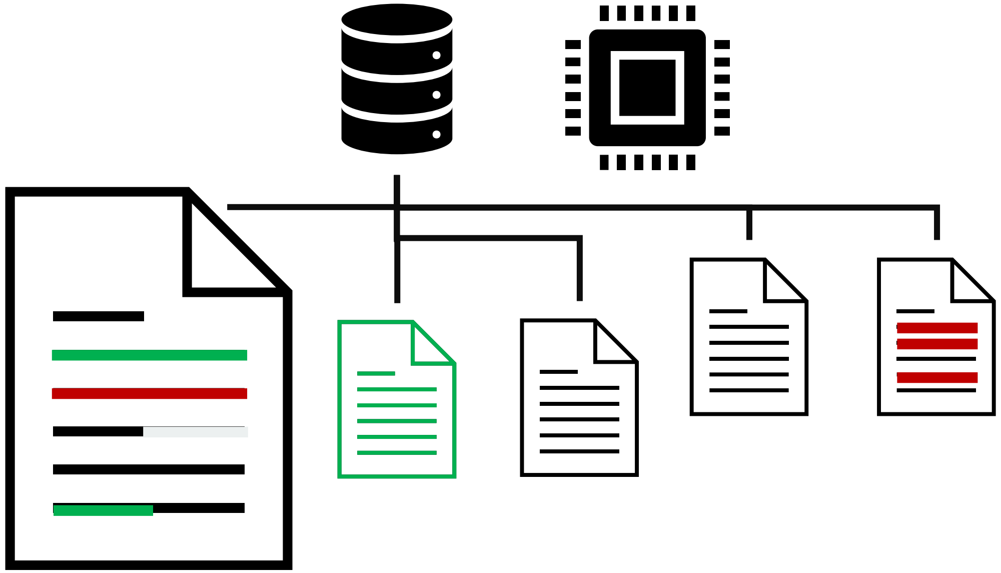
Five Challenges for "AI for Code"

1. Expanding Context
2. Software Maintenance



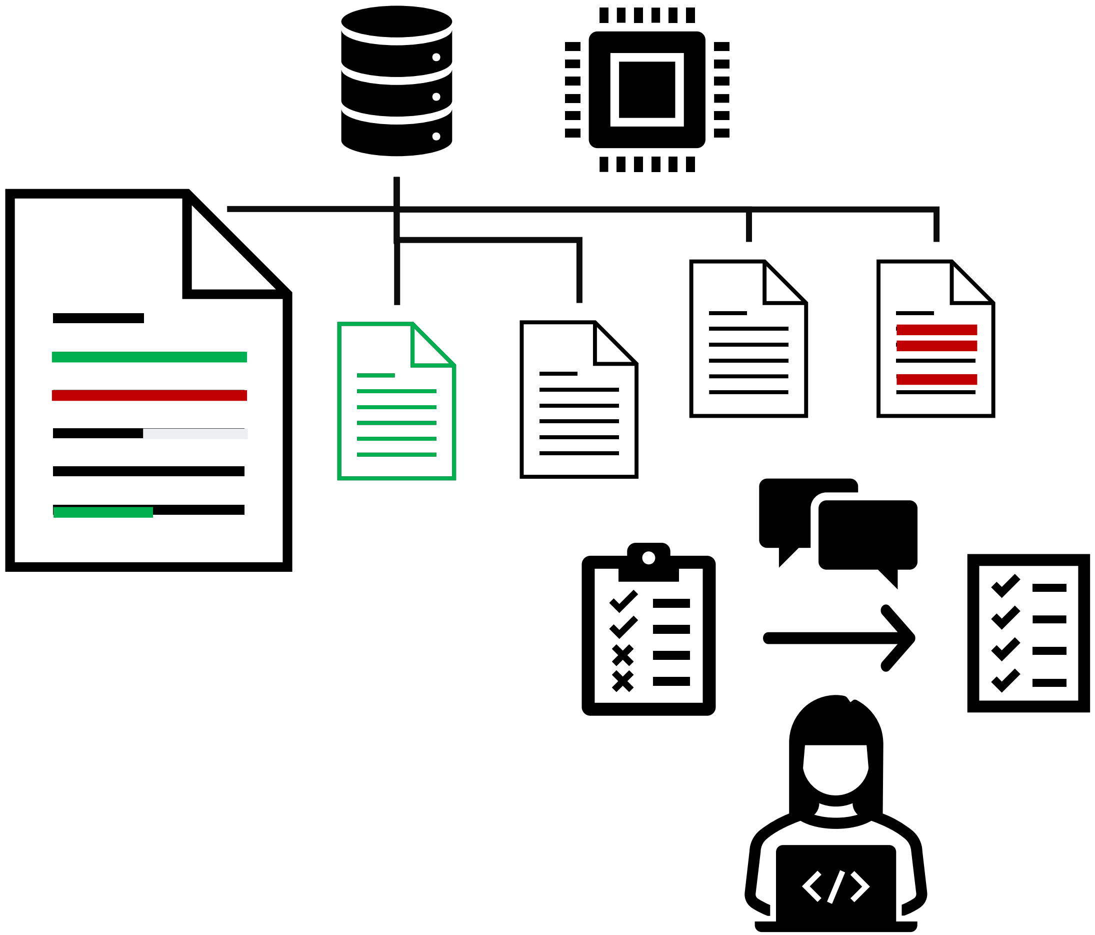
Five Challenges for “AI for Code”

1. Expanding Context
2. Software Maintenance
3. Modeling Semantics



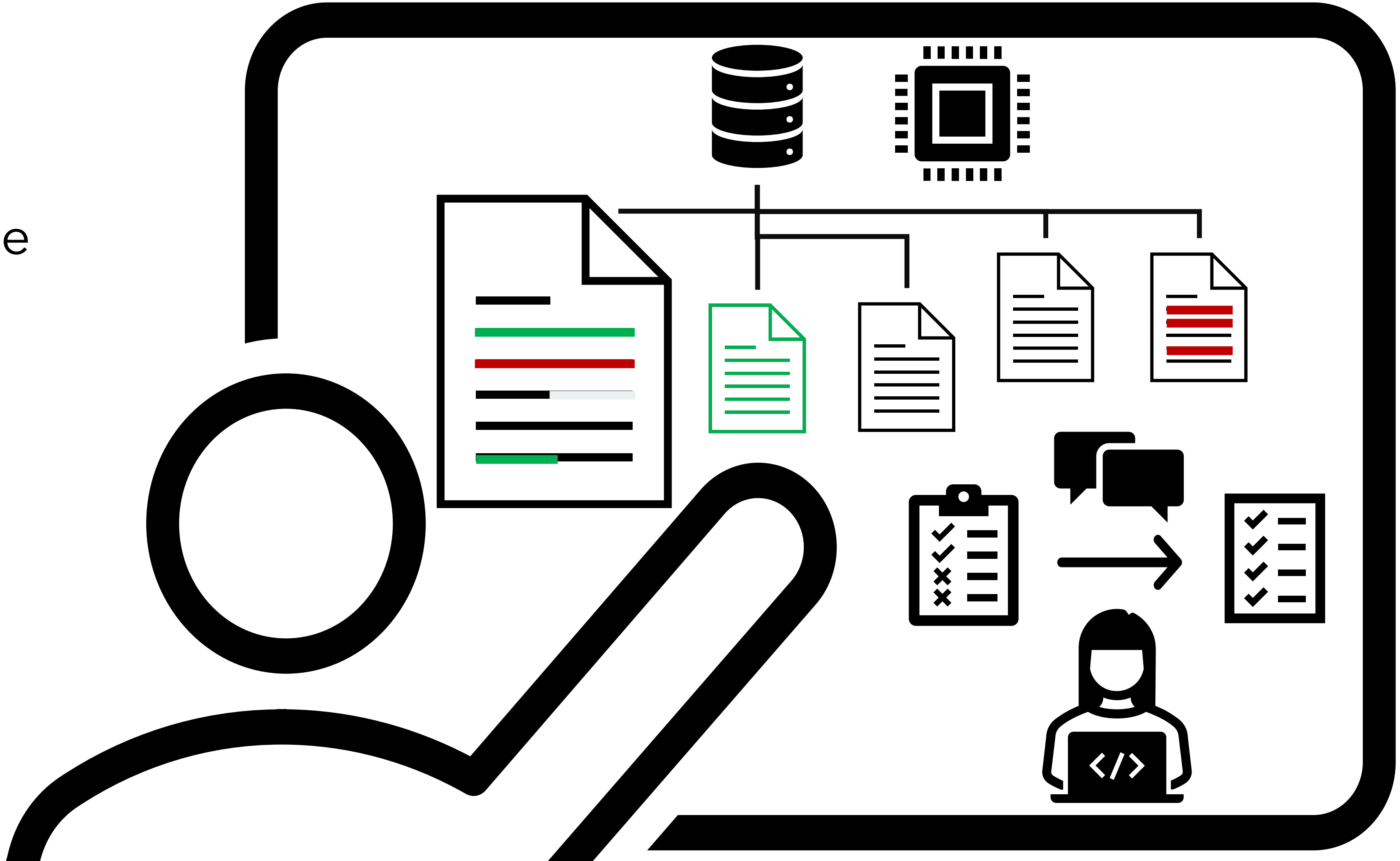
Five Challenges for “AI for Code”

1. Expanding Context
2. Software Maintenance
3. Modeling Semantics
4. Interacting in Teams



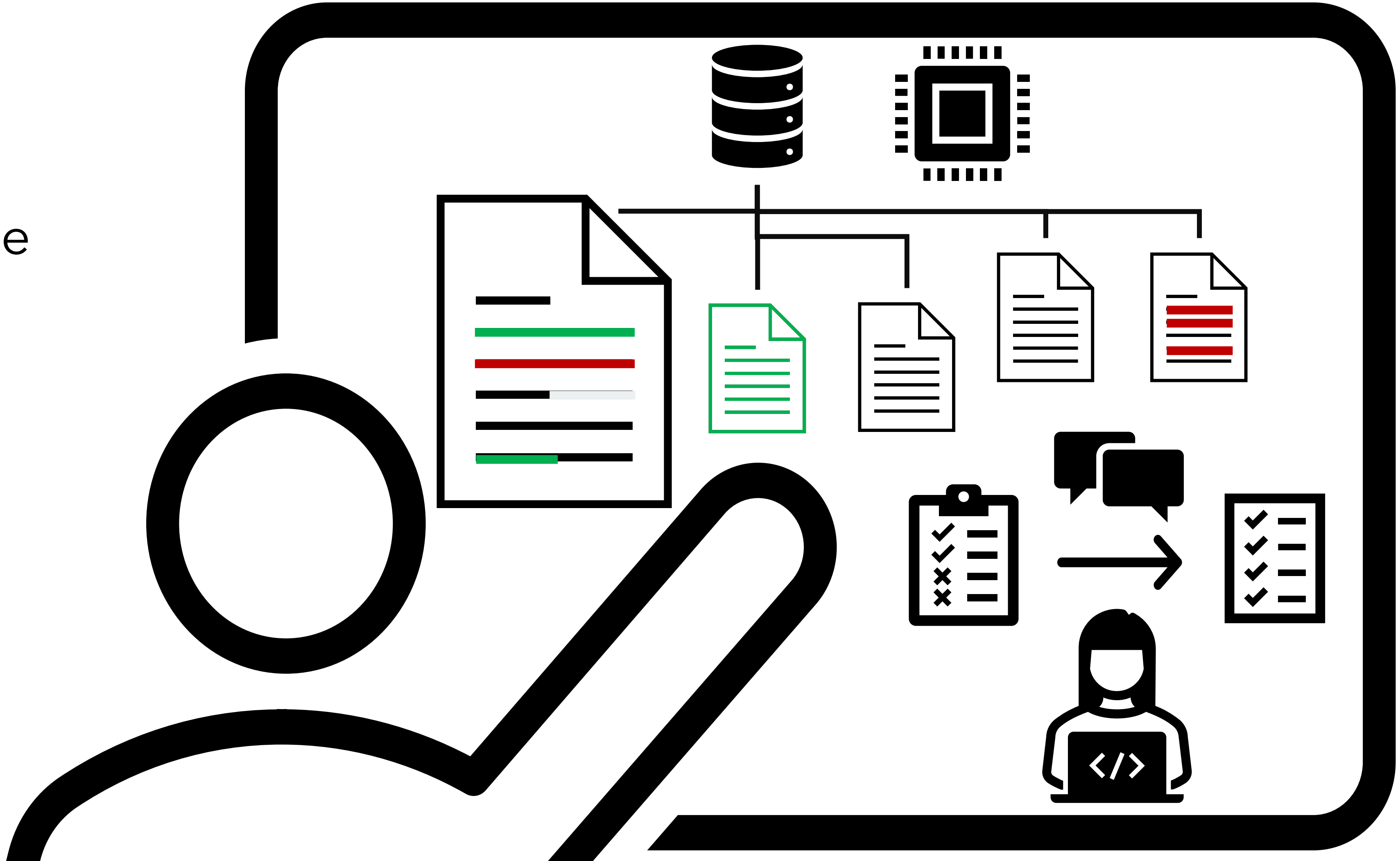
Five Challenges for “AI for Code”

1. Expanding Context
2. Software Maintenance
3. Modeling Semantics
4. Interacting in Teams
5. Navigating Process



Five Challenges for "AI for Code"

1. **Expanding Context**
2. Software Maintenance
3. Modeling Semantics
4. Interacting in Teams
5. Navigating Process



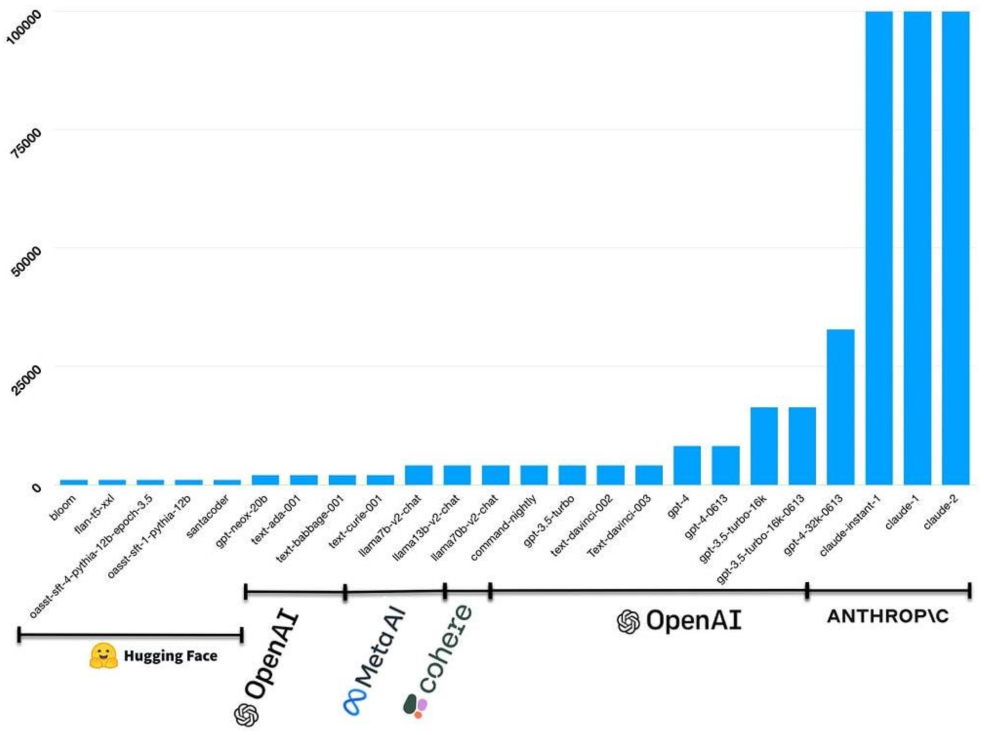
Context Scaling

LLM contexts have **grown rapidly**

Isn't attention cost **quadratic in input length**?

A: Who cares

- It's a **small fraction** of overall compute
- Often **worth the cost** during inference



https://cobusgreyling.medium.com/rag-llm-context-size-6728a2f44beb

Context Scaling

Context is what you **make of it**

- Pretraining "packs" multiples files together
- Shuffle GitHub? The model learns to ignore adjacent files

code_foo.py

code_bar.py

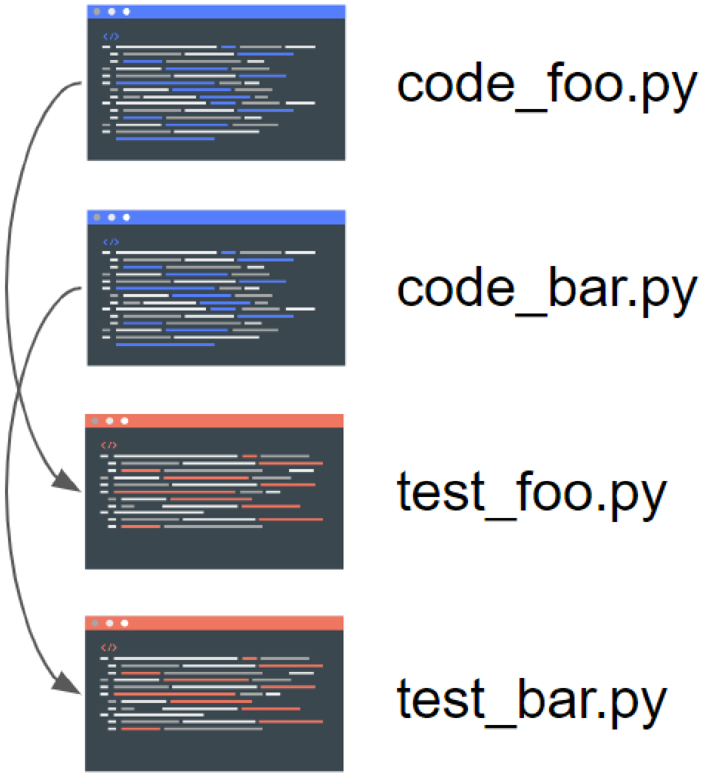
test_foo.py

test_bar.py

Context Scaling

Context is what you **make of it**

- Pretraining "packs" multiples files together
- Shuffle GitHub? The model learns to ignore adjacent files
- Which is a waste



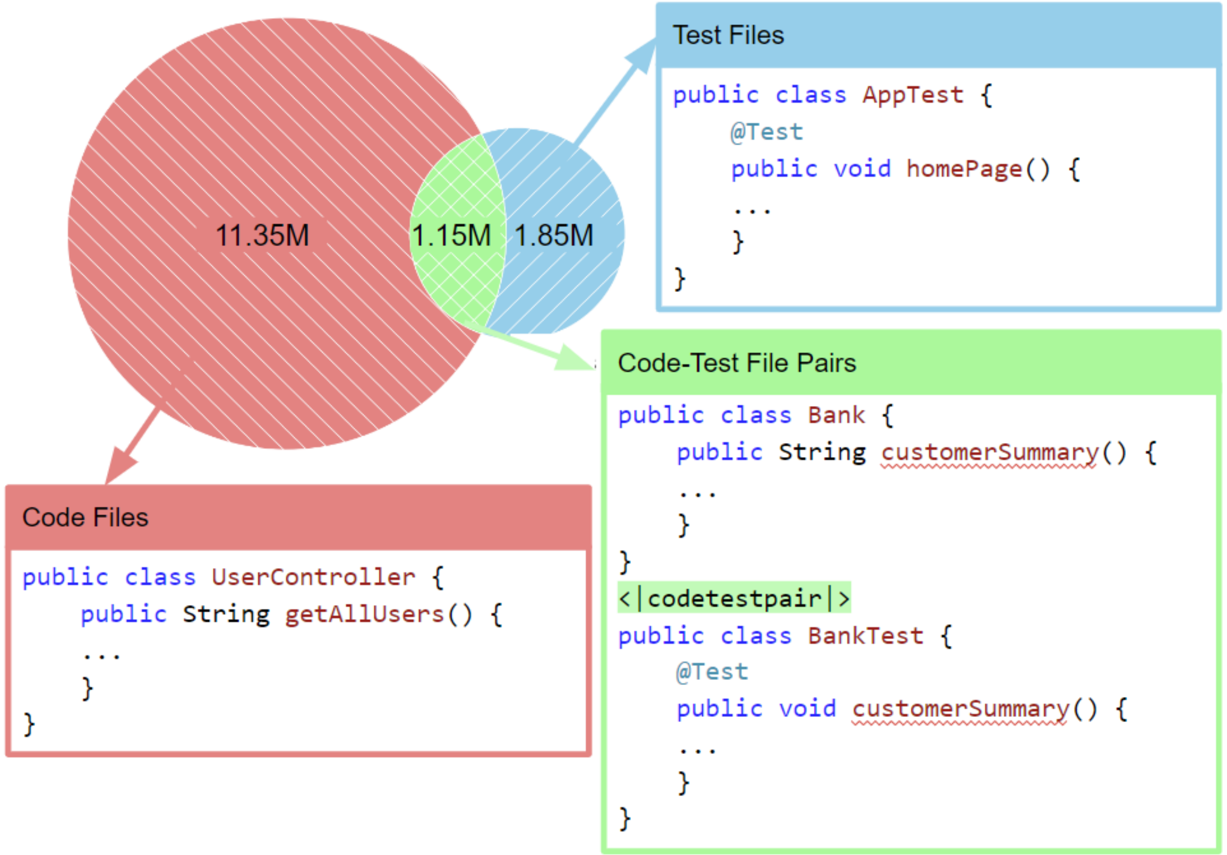
Rao et al., ASE'23. CAT-LM 😺Training Language Models on Aligned Code And Tests

Using Long Contexts: Test Prediction

Finding relevant data

Only 1M code/test pairs on GitHub?

No problem! Just **use everything**



Test Files

```
public class AppTest {
    @Test
    public void homePage() {
    ...
    }
}
```

Code-Test File Pairs

```
public class Bank {
    public String customerSummary() {
    ...
    }
}
<|codetestpair|>
public class BankTest {
    @Test
    public void customerSummary() {
    ...
    }
}
```

Code Files

```
public class UserController {
    public String getAllUsers() {
    ...
    }
}
```

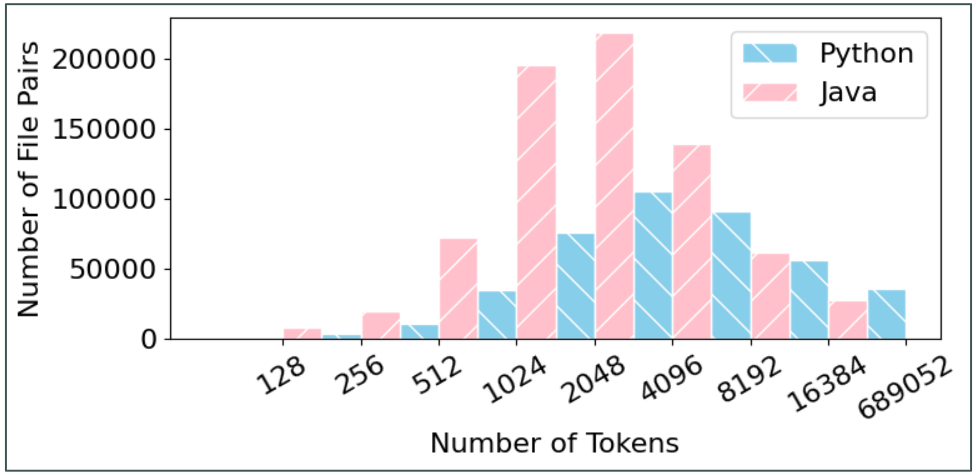
Rao et al., ASE'23. CAT-LM 😺Training Language Models on Aligned Code And Tests

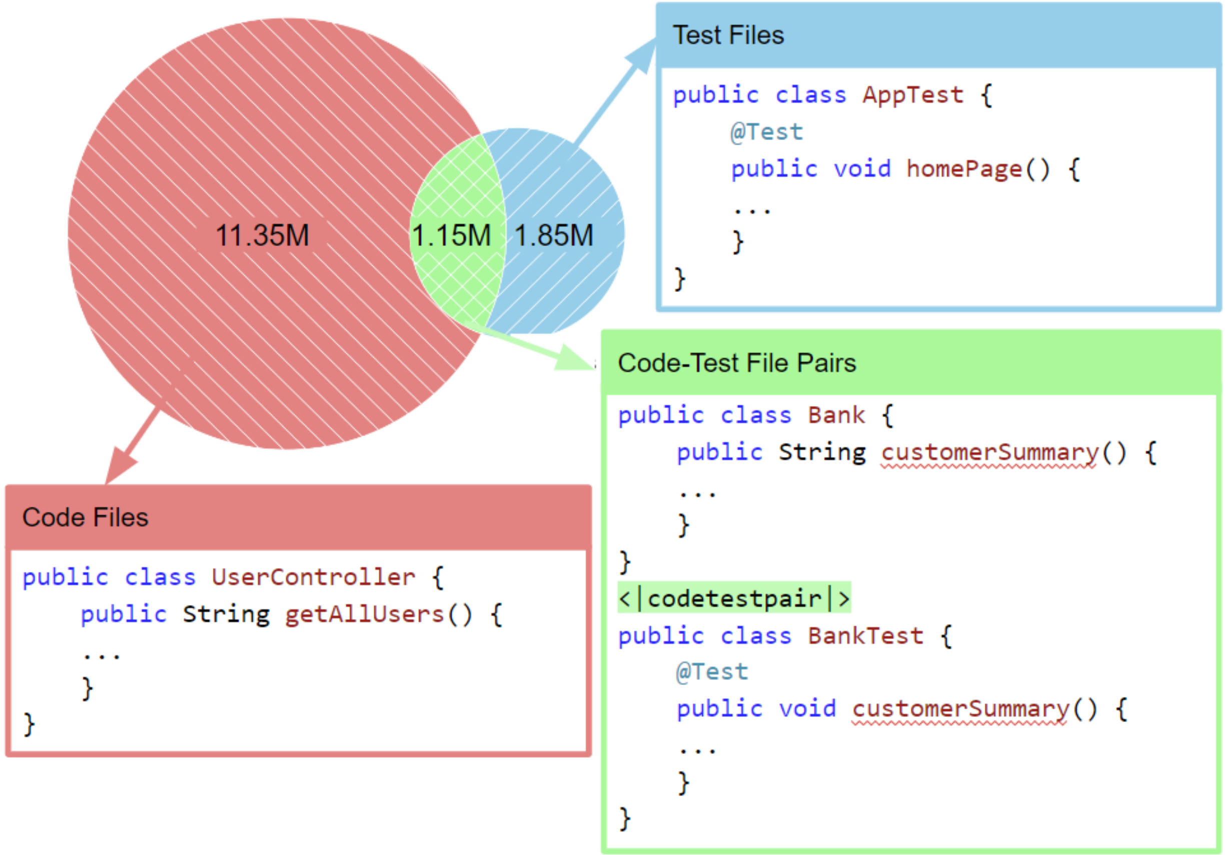
Using Long Contexts:
Test Prediction

Finding relevant data

Some files are very long?

Throw **compute** at it!





Test Files

```
public class AppTest {
    @Test
    public void homePage() {
      ...
    }
}
```

Code-Test File Pairs

```
public class Bank {
    public String customerSummary() {
      ...
    }
}
<|codetestpair|>
public class BankTest {
    @Test
    public void customerSummary() {
      ...
    }
}
```

Code Files

```
public class UserController {
    public String getAllUsers() {
      ...
    }
}
```

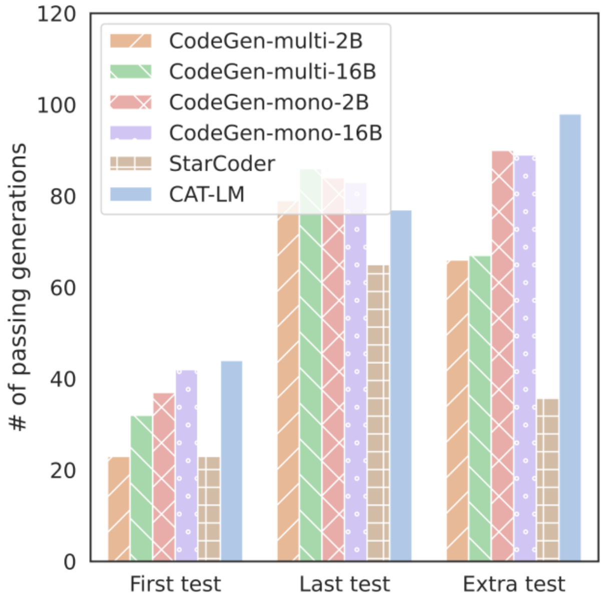
Rao et al., ASE’23. CAT-LM 🐱 Training Language Models on Aligned Code And Tests

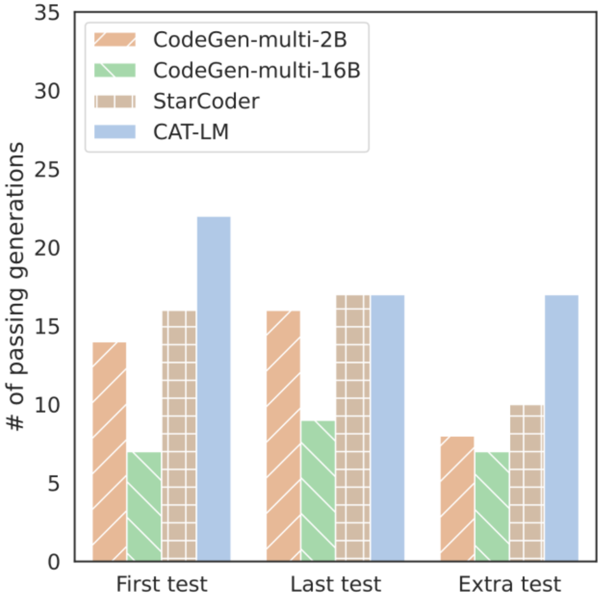
Using Long Contexts: Test Prediction

Of course this helps

We generate **more valid tests** on a fraction of the budget

- Coverage approaches, but **doesn't quite match**, human programmers
- Larger models could do better





Work led by Nikitha Rao (nikitharao@cmu.edu)
Cloud compute credits contributed by Google

Rao et al., ASE'23. CAT-LM 🐱 Training Language Models on Aligned Code And Tests

Context Scaling – Lessons

Spend **compute** where it is due

Scaling **parameters** and **data** are slowing down. **Context** has a lot to offer.

Add **value through data**

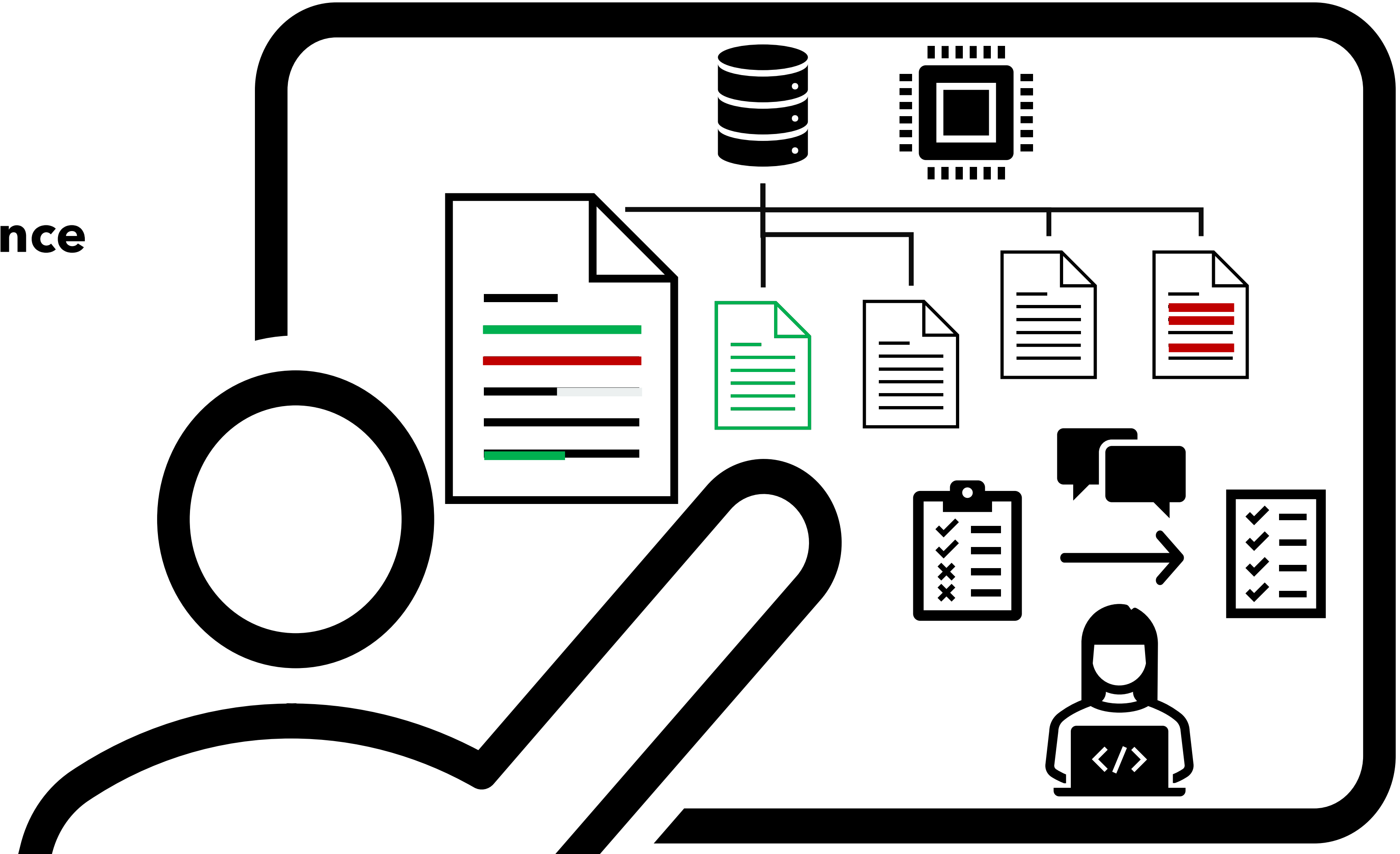
Better a **million good tokens** than a trillion boring ones

Still, a **long way to go** for modeling real contexts

See SWEBench

Five Challenges for "AI for Code"

1. Expanding Context
2. **Software Maintenance**
3. Modeling Semantics
4. Interacting in Teams
5. Navigating Process



Software Maintenance

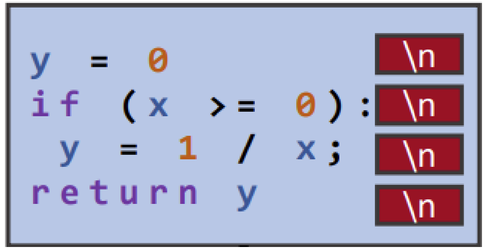
Developers don’t only Write Code

Test prediction is still **generative**. What about **understanding, analyzing** code?

Software Maintenance: Fault Localization

A Case Study

All code has **bugs**, but most code isn't buggy.



```
y = 0
if (x >= 0):
 y = 1 / x;
return y
```

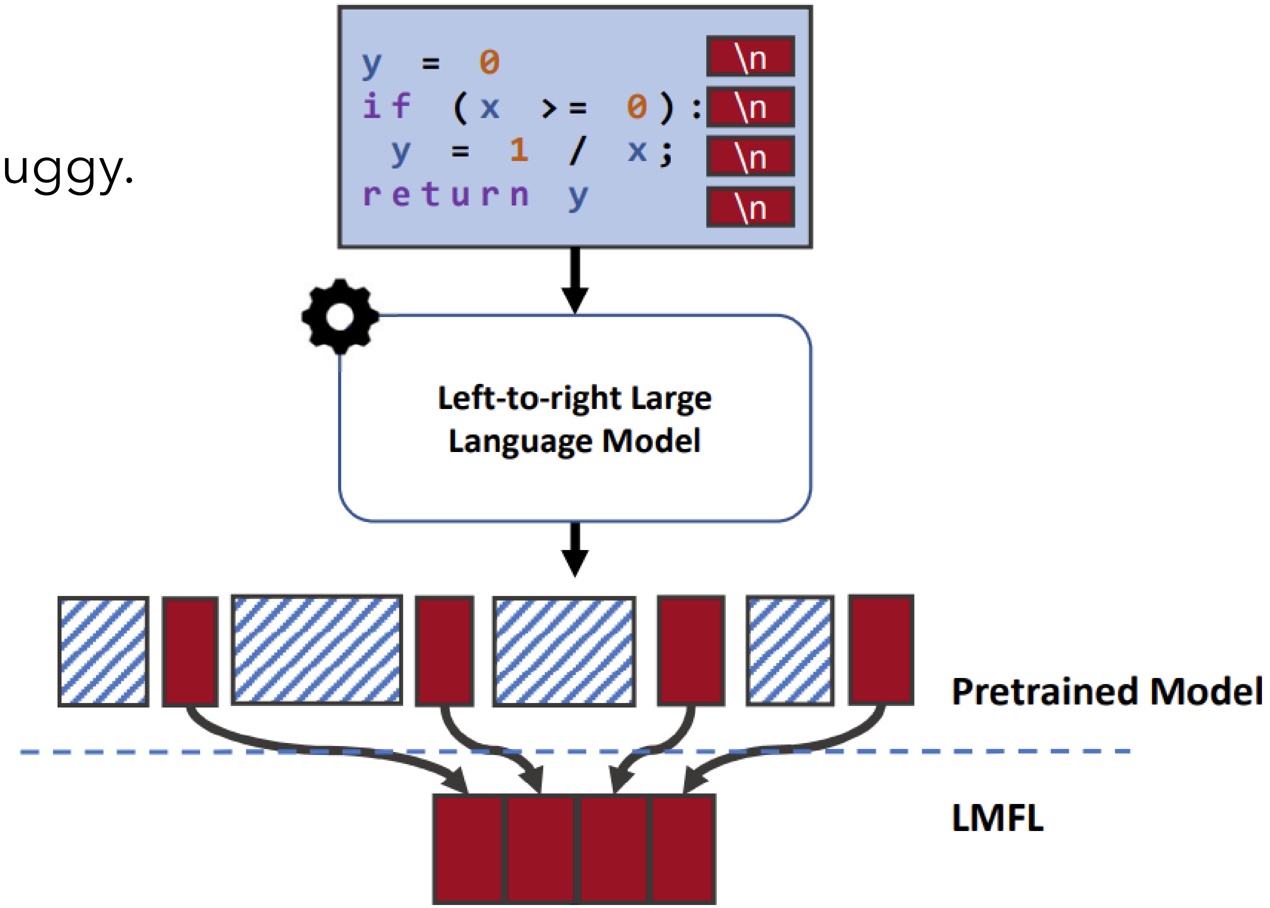
Yang et al., ICSE'24. Large Language Models for Test-Free Fault Localization. https://arxiv.org/abs/2310.01726

Software Maintenance: Fault Localization

A Case Study

All code has **bugs**, but most code isn't buggy.

LLM have read all code. Can they tell?



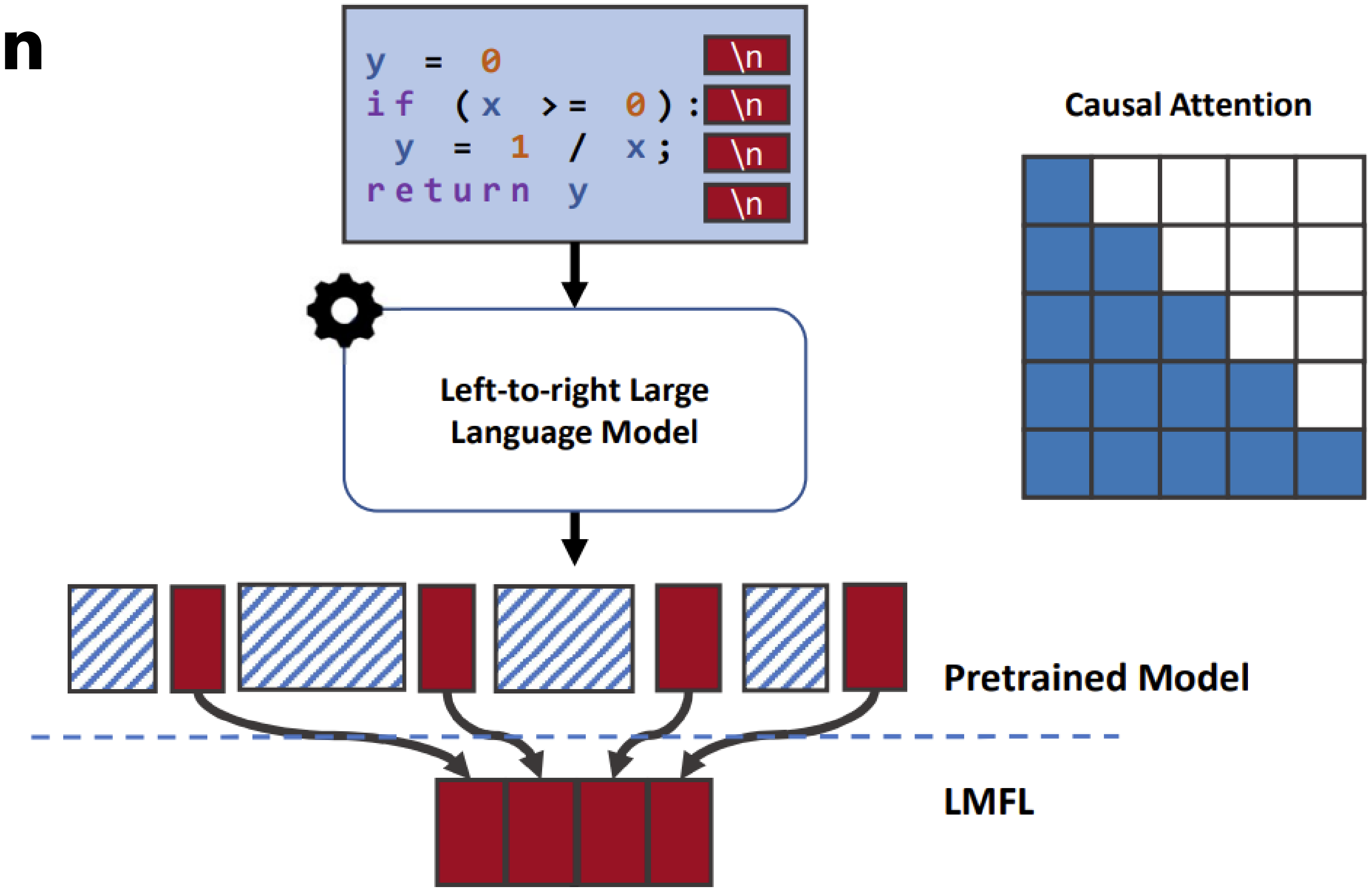
```
y = 0                 \n
if (x >= 0):          \n
  y = 1 / x;          \n
return y              \n
```

Yang et al., ICSE'24. Large Language Models for Test-Free Fault Localization. https://arxiv.org/abs/2310.01726

Software Maintenance: Fault Localization

Generation ≠ Interpretation

But, the information **is already there**



```
y = 0
if (x >= 0):
 y = 1 / x;
return y
```

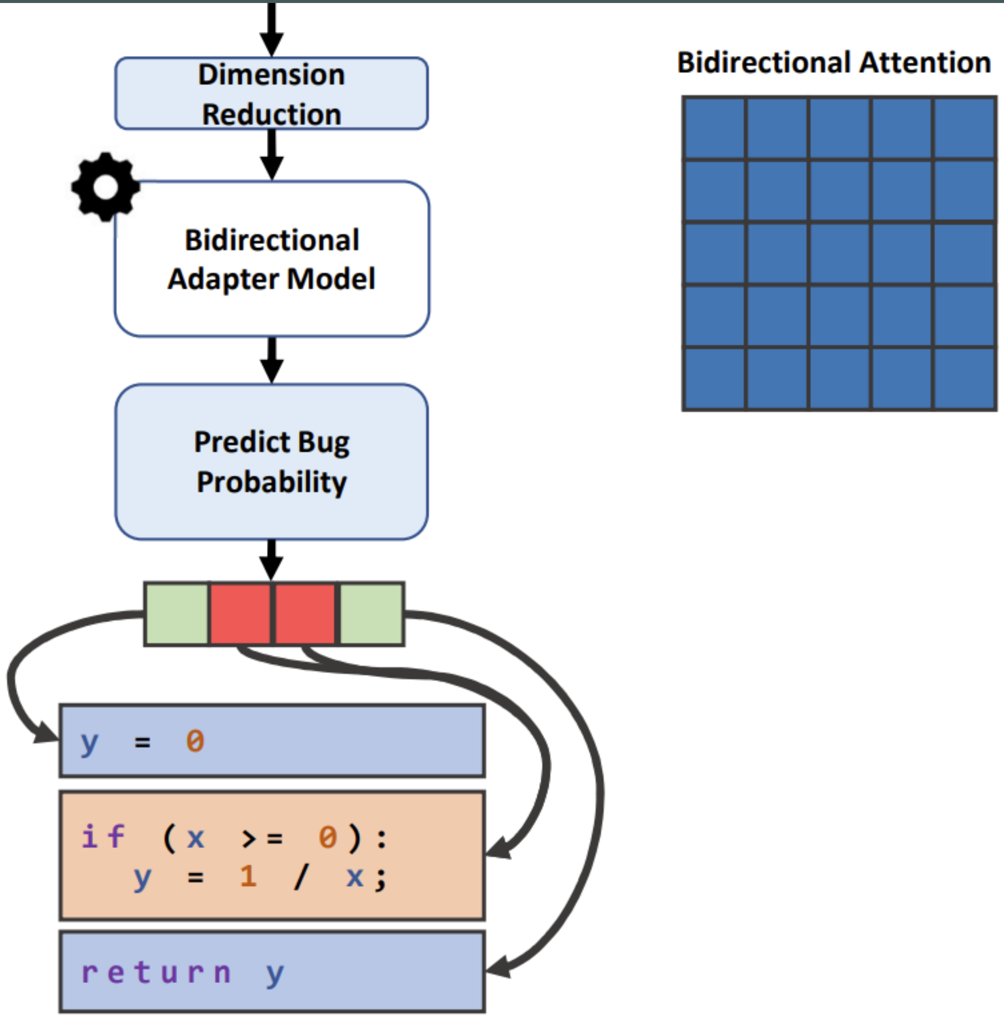
Yang et al., ICSE'24. Large Language Models for Test-Free Fault Localization. https://arxiv.org/abs/2310.01726

Software Maintenance: Fault Localization

Generation ≠ Interpretation

In this work, we train:

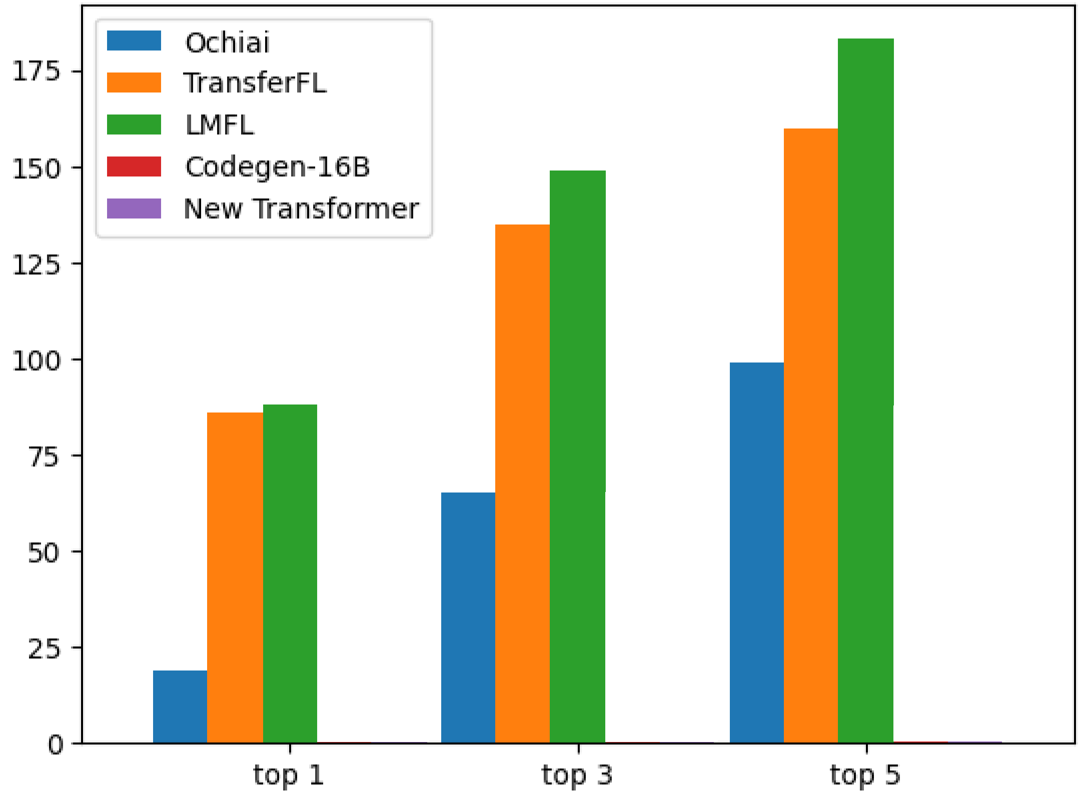
- Low-dimensional, bidirectional **adapter layers**
- A lightweight classifier



Yang et al., ICSE’24. Large Language Models for Test-Free Fault Localization. https://arxiv.org/abs/2310.01726

Software Maintenance: Fault Localization

Improves over **purpose-built** methods



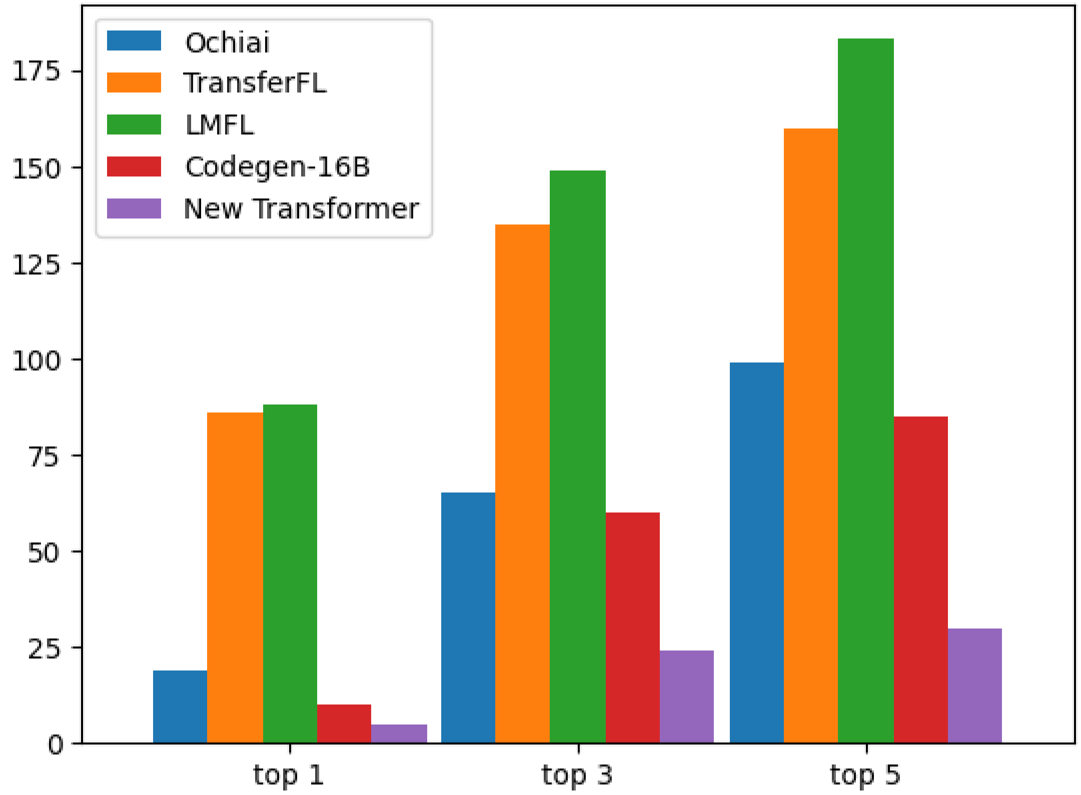
Work led by Aidan Yang (aidan@cmu.edu)

Yang et al., ICSE'24. Large Language Models for Test-Free Fault Localization. https://arxiv.org/abs/2310.01726

Software Maintenance: Fault Localization

Improves over **purpose-built** methods

Drastically improves over the **initial LLM**



Work led by Aidan Yang (aidan@cmu.edu)

Yang et al., ICSE'24. Large Language Models for Test-Free Fault Localization. https://arxiv.org/abs/2310.01726

A Theme Emerges

Lean on the **pretrained model**

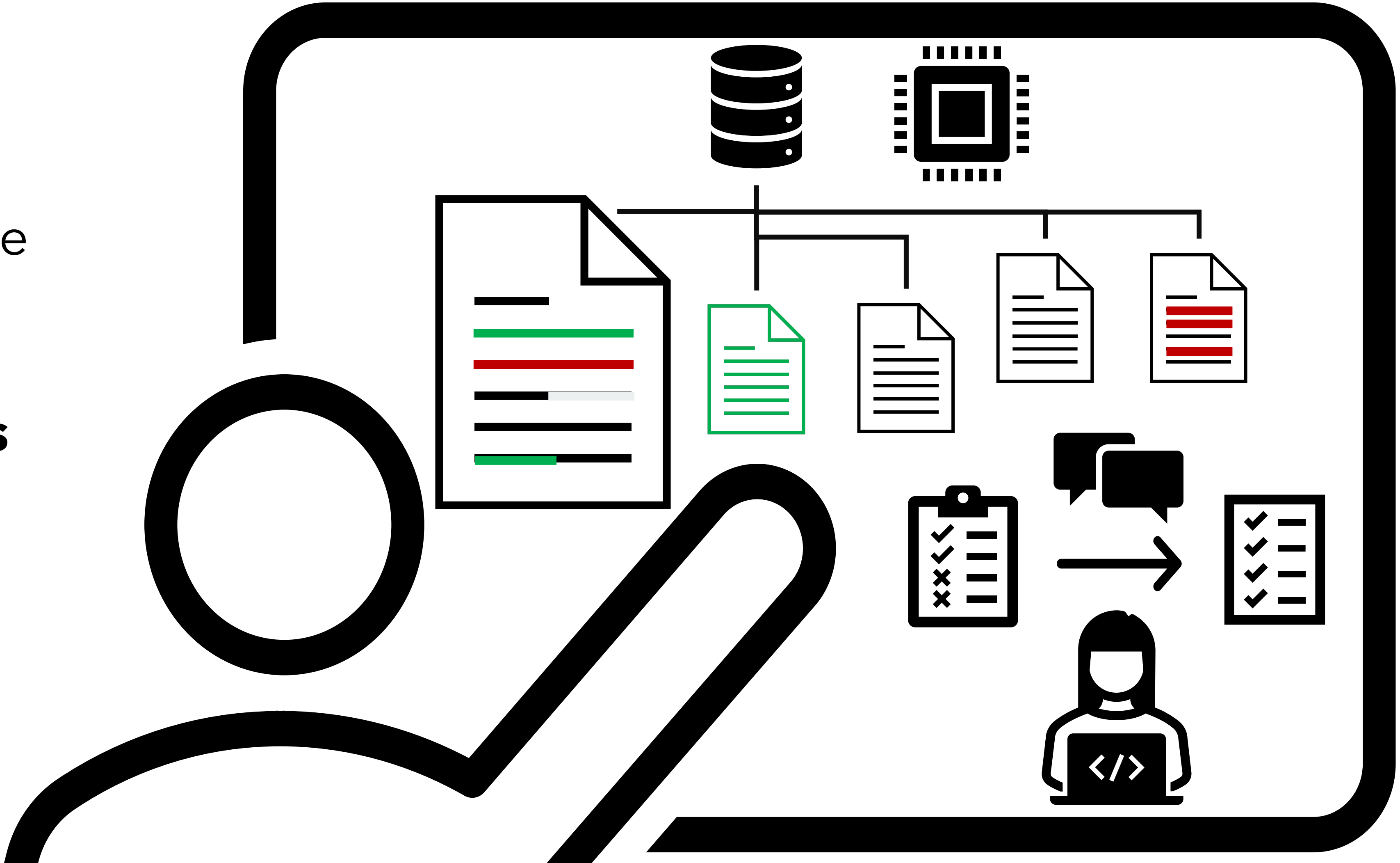
LLMs have seen **10,000x** what we can read in a lifetime

A ton of knowledge is **untapped** in those weights

To **tap in**, research needs to leverage on SE knowledge

Five Challenges for “AI for Code”

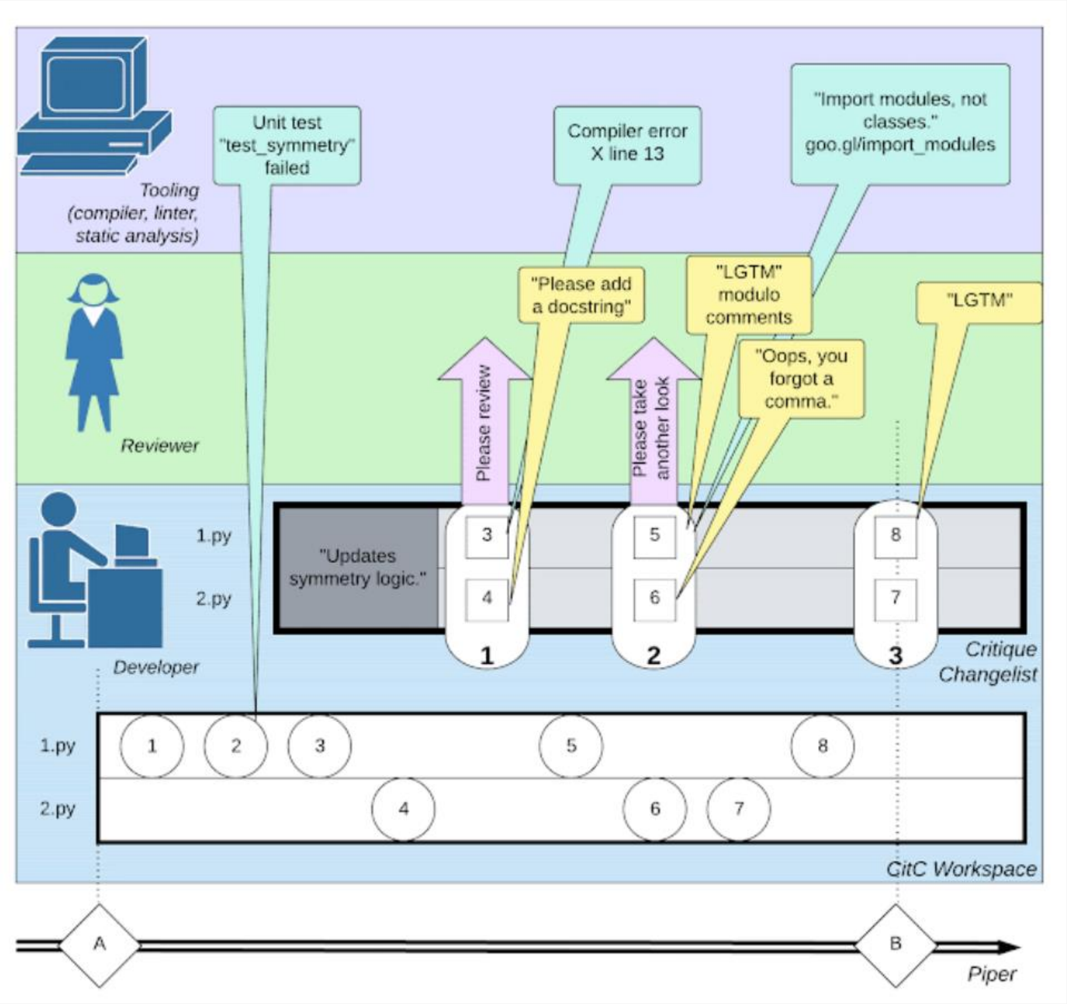
1. Expanding Context
2. Software Maintenance
3. ~~Modeling Semantics~~
4. **Interacting in Teams**
5. Navigating Process



Modeling the SE Process

Software development involves a lot **more than coding.**

Can we model the whole process?



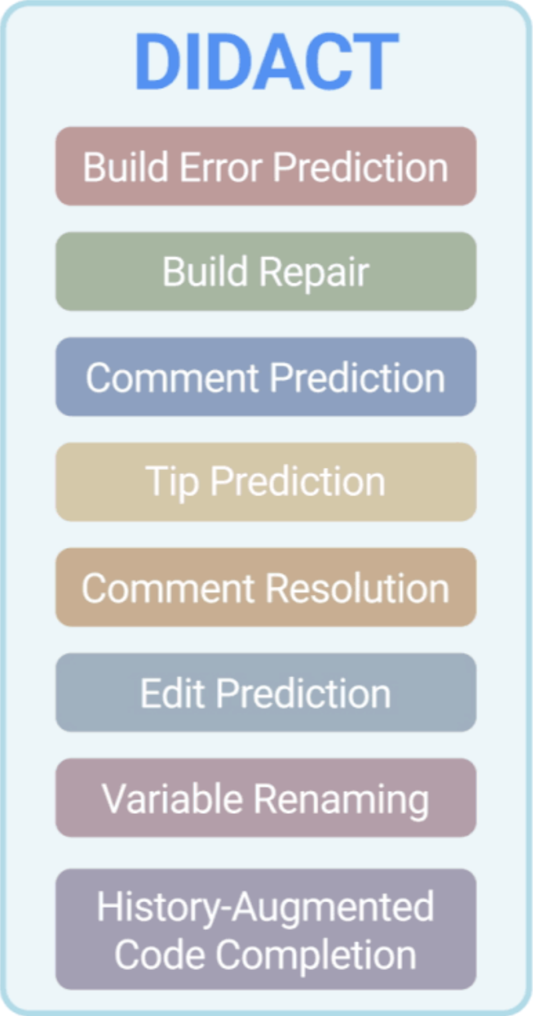
https://blog.research.google/2023/05/large-sequence-models-for-software.html

Modeling the SE Process

A more **holistic** view

Many tasks are **connected**

Developers appreciate help anywhere, but **good UX is key**



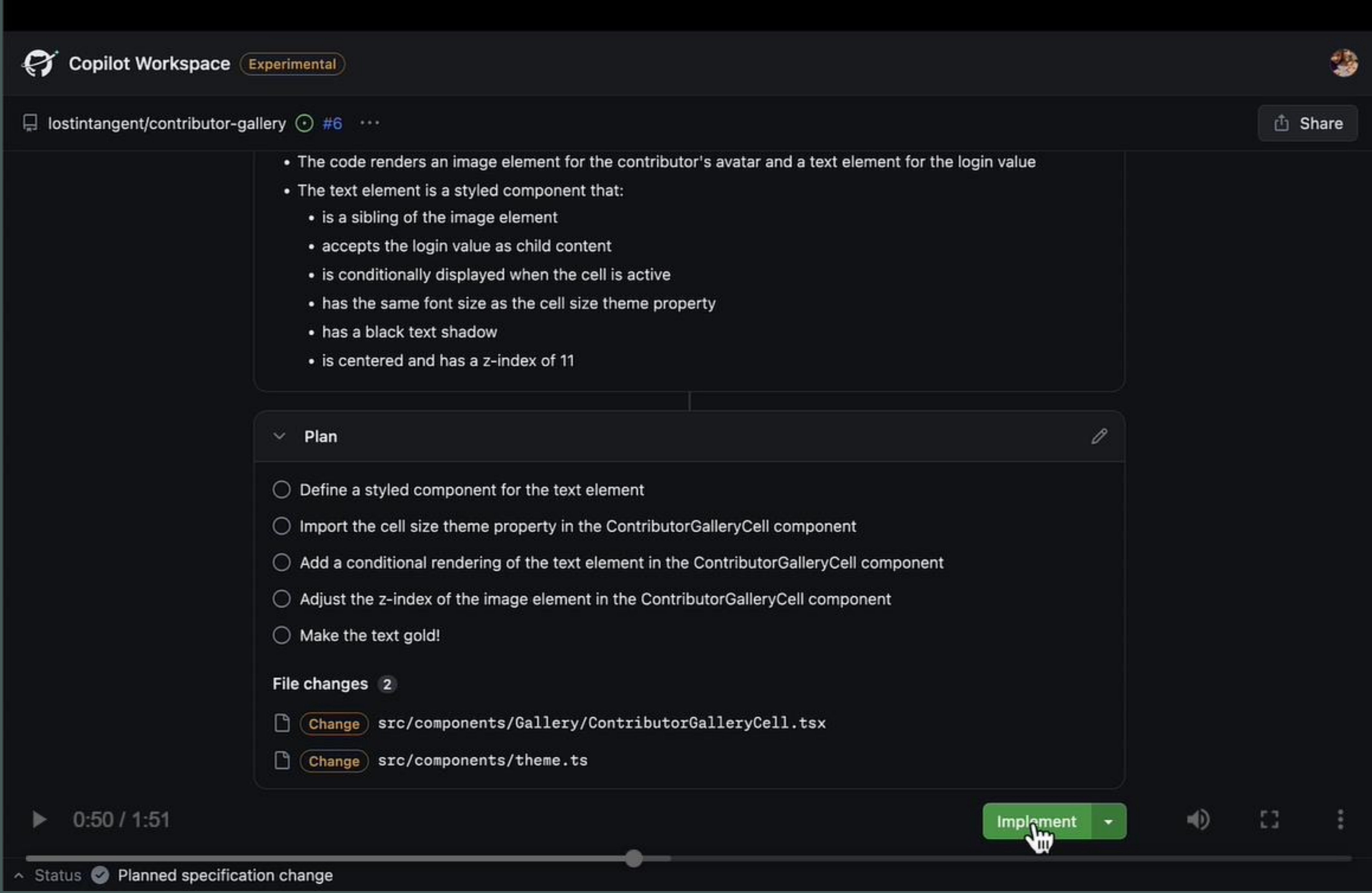
DIDACT

- Build Error Prediction
- Build Repair
- Comment Prediction
- Tip Prediction
- Comment Resolution
- Edit Prediction
- Variable Renaming
- History-Augmented Code Completion

https://blog.research.google/2023/05/large-sequence-mode

Modeling the SE Process

A few more examples of **promising trends**



- The code renders an image element for the contributor's avatar and a text element for the login value
- The text element is a styled component that:
  - is a sibling of the image element
  - accepts the login value as child content
  - is conditionally displayed when the cell is active
  - has the same font size as the cell size theme property
  - has a black text shadow
  - is centered and has a z-index of 11

Plan

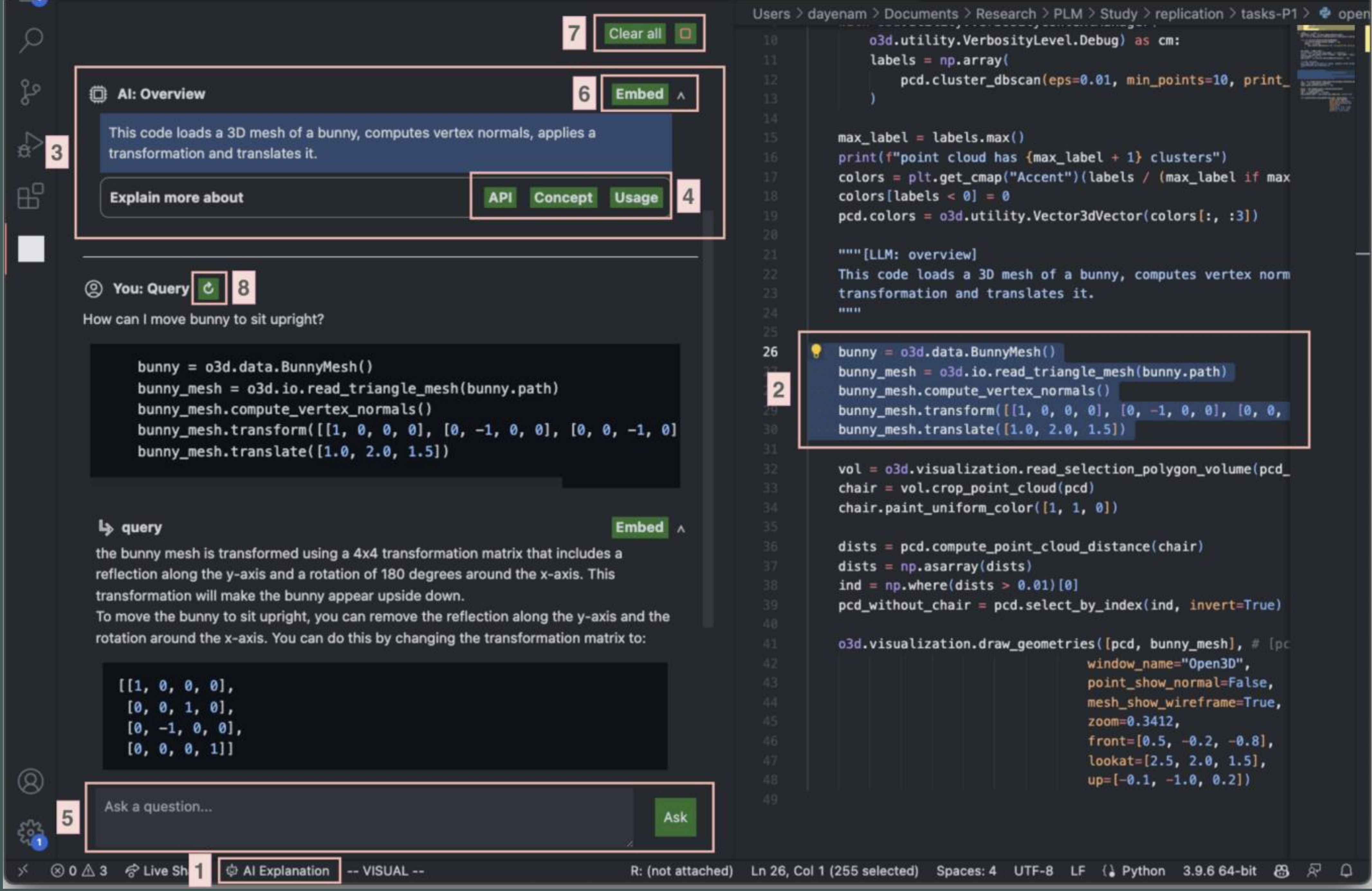
○ Define a styled component for the text element
○ Import the cell size theme property in the ContributorGalleryCell component
○ Add a conditional rendering of the text element in the ContributorGalleryCell component
○ Adjust the z-index of the image element in the ContributorGalleryCell component
○ Make the text gold!

File changes 2

Change `src/components/Gallery/ContributorGalleryCell.tsx`
Change `src/components/theme.ts`

Status Planned specification change

https://githubnext.com/projects/copilot-workspace/



AI: Overview

This code loads a 3D mesh of a bunny, computes vertex normals, applies a transformation and translates it.

Explain more about

You: Query

How can I move bunny to sit upright?

```
bunny = o3d.data.BunnyMesh()
bunny_mesh = o3d.io.read_triangle_mesh(bunny.path)
bunny_mesh.compute_vertex_normals()
bunny_mesh.transform([[1, 0, 0, 0], [0, -1, 0, 0], [0, 0, -1, 0]
bunny_mesh.translate([1.0, 2.0, 1.5])
```

↳ **query**

the bunny mesh is transformed using a 4x4 transformation matrix that includes a reflection along the y-axis and a rotation of 180 degrees around the x-axis. This transformation will make the bunny appear upside down.
To move the bunny to sit upright, you can remove the reflection along the y-axis and the rotation around the x-axis. You can do this by changing the transformation matrix to:

```
[[1, 0, 0, 0],
 [0, 0, 1, 0],
 [0, -1, 0, 0],
 [0, 0, 0, 1]]
```

Ask a question...

```
        o3d.utility.VerbosityLevel.Debug) as cm:
        labels = np.array(
            pcd.cluster_dbscan(eps=0.01, min_points=10, print_
        )

    max_label = labels.max()
    print(f"point cloud has {max_label + 1} clusters")
    colors = plt.get_cmap("Accent")(labels / (max_label if max
    colors[labels < 0] = 0
    pcd.colors = o3d.utility.Vector3dVector(colors[:, :3])

    """[LLM: overview]
    This code loads a 3D mesh of a bunny, computes vertex norm
    transformation and translates it.
    """

    bunny = o3d.data.BunnyMesh()
    bunny_mesh = o3d.io.read_triangle_mesh(bunny.path)
    bunny_mesh.compute_vertex_normals()
    bunny_mesh.transform([[1, 0, 0, 0], [0, -1, 0, 0], [0, 0,
    bunny_mesh.translate([1.0, 2.0, 1.5])

    vol = o3d.visualization.read_selection_polygon_volume(pcd_
    chair = vol.crop_point_cloud(pcd)
    chair.paint_uniform_color([1, 1, 0])

    dists = pcd.compute_point_cloud_distance(chair)
    dists = np.asarray(dists)
    ind = np.where(dists > 0.01)[0]
    pcd_without_chair = pcd.select_by_index(ind, invert=True)

    o3d.visualization.draw_geometries([pcd, bunny_mesh], # [pc
                                      window_name="Open3D",
                                      point_show_normal=False,
                                      mesh_show_wireframe=True,
                                      zoom=0.3412,
                                      front=[0.5, -0.2, -0.8],
                                      lookat=[2.5, 2.0, 1.5],
                                      up=[-0.1, -1.0, 0.2])
```

Nam et al., In-IDE Generation-based Information Support with a Large Language Model. ICSE'24. https://arxiv.org/abs/2307.08177

A Call to Action

The field needs **concerted efforts** to study AI in the wild

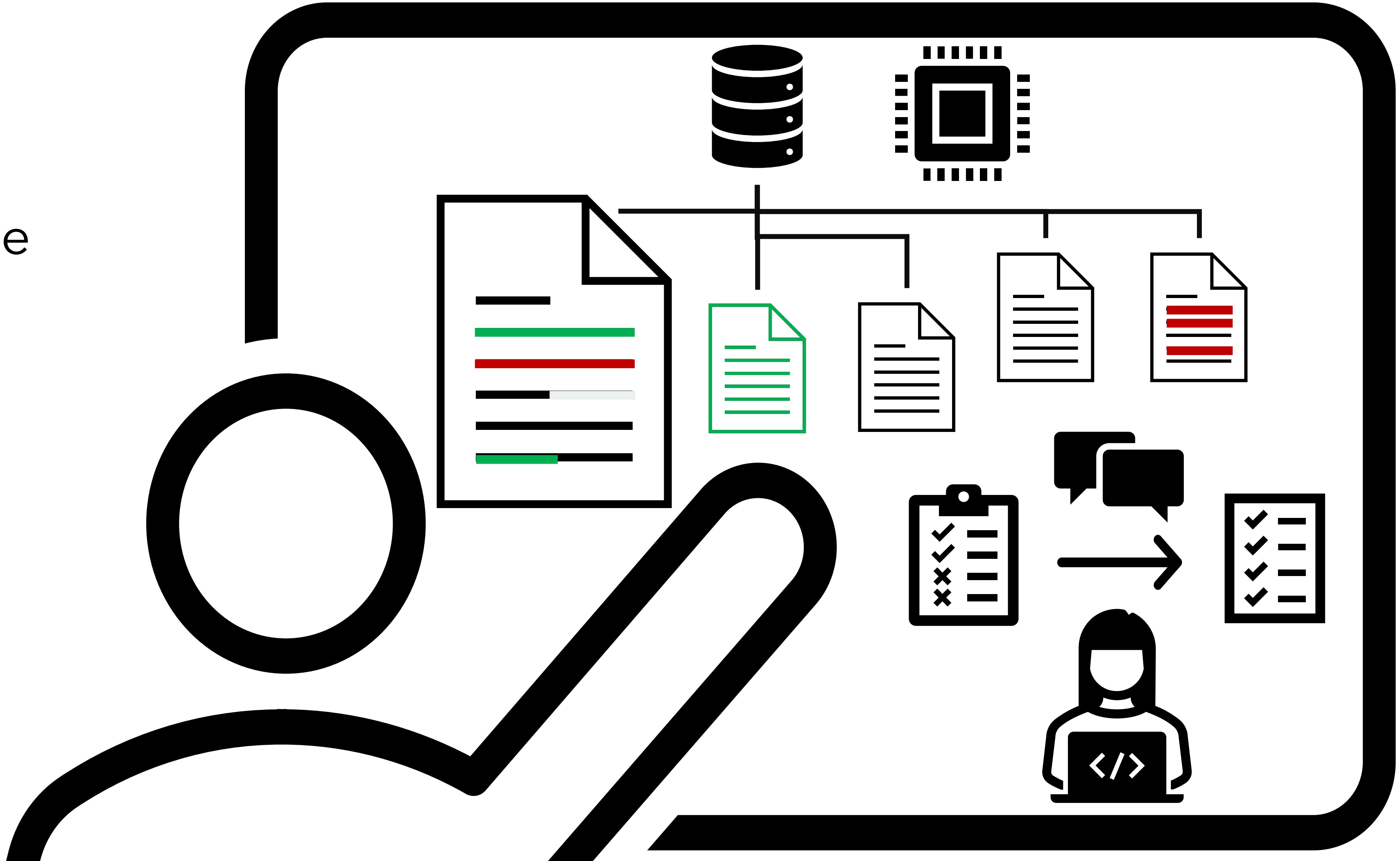
AI is inherently **about people** – observational studies are becoming key

LLMs have entered **supercomputing territory** – no lab can do this alone

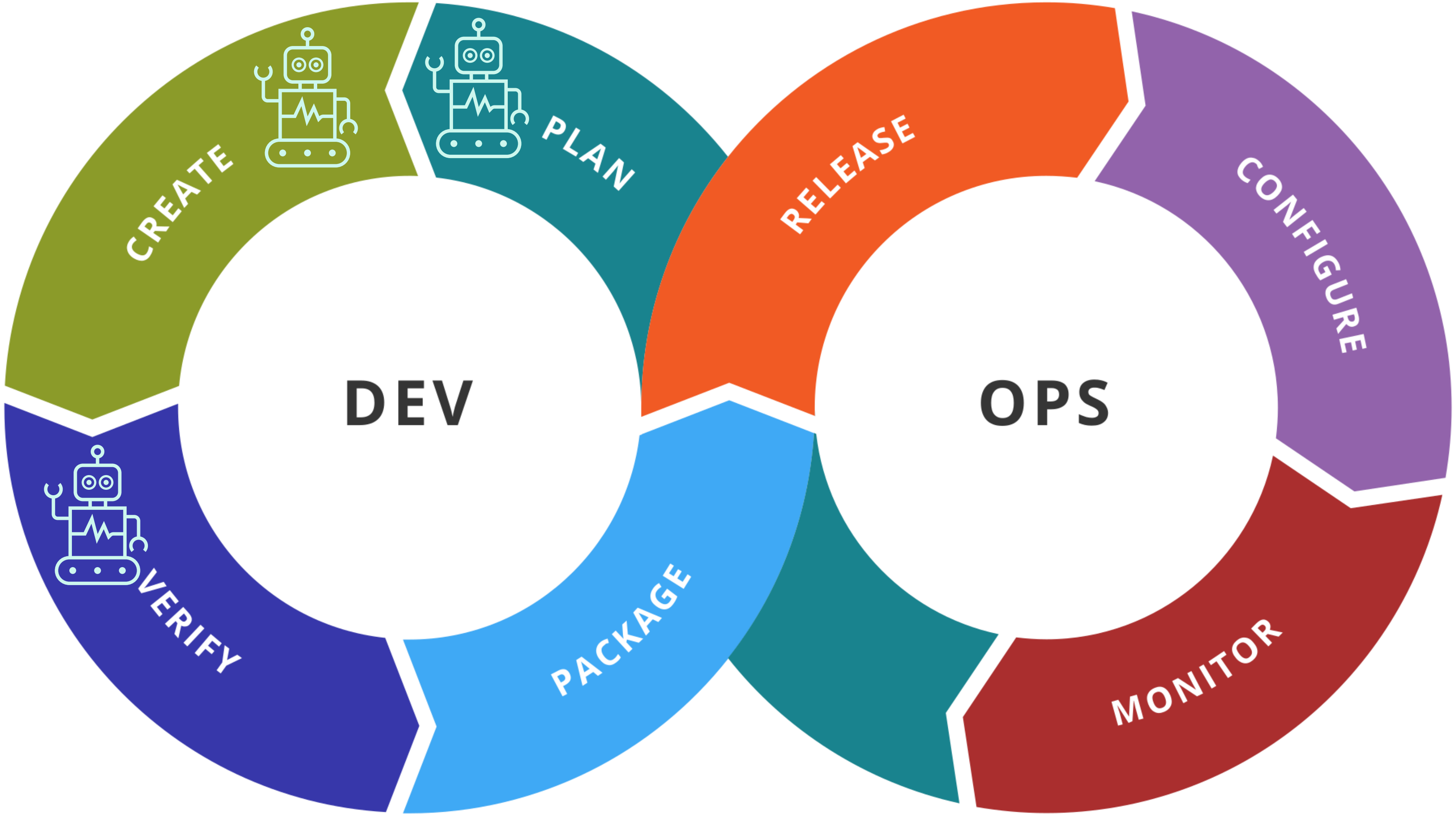
Come together and build **high-quality benchmarks & tools**

Five Challenges for "AI for Code"

1. Expanding Context
2. Software Maintenance
3. Modeling Semantics
4. Interacting in Teams
5. **Navigating Process**

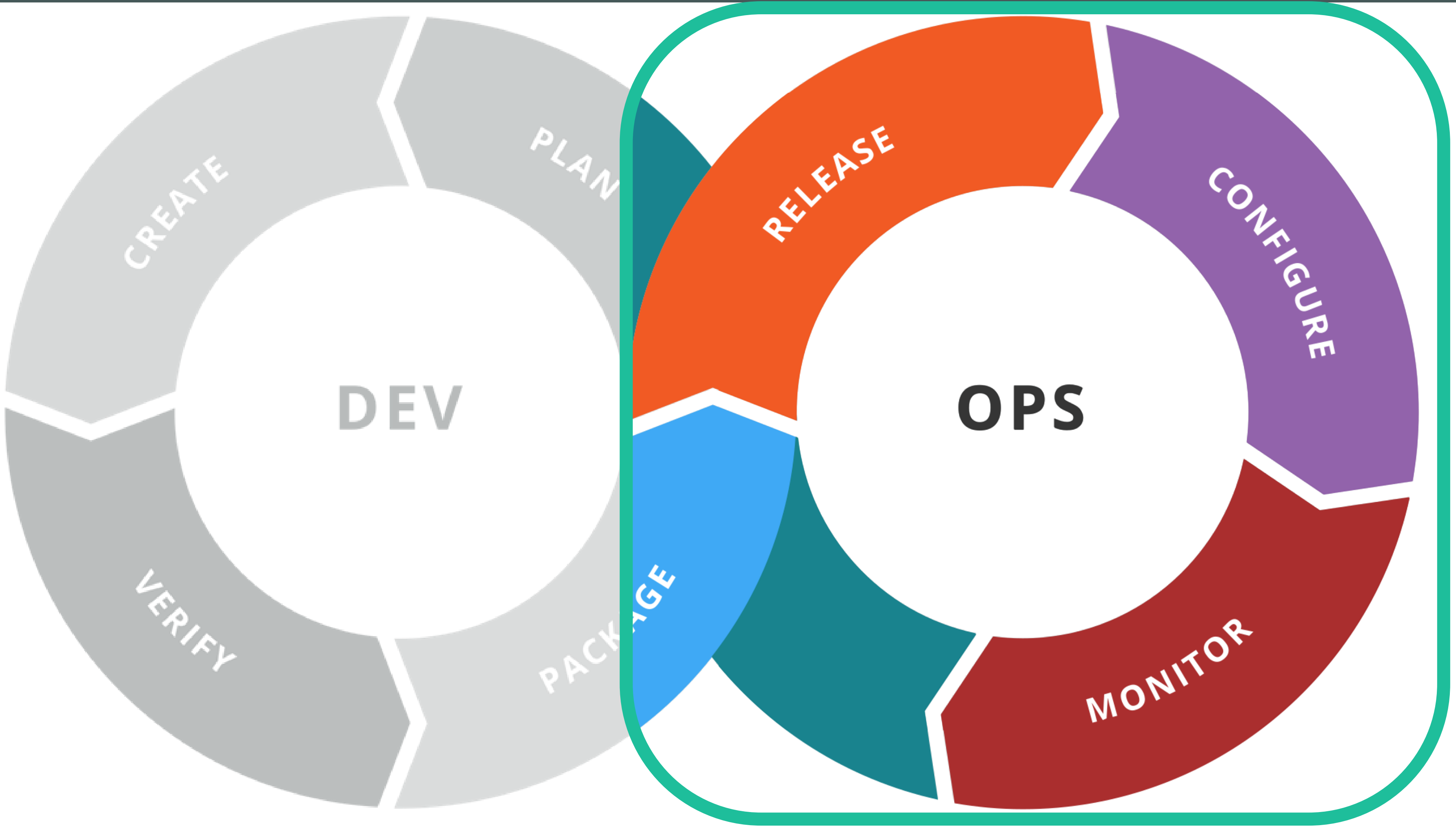


What's Next?



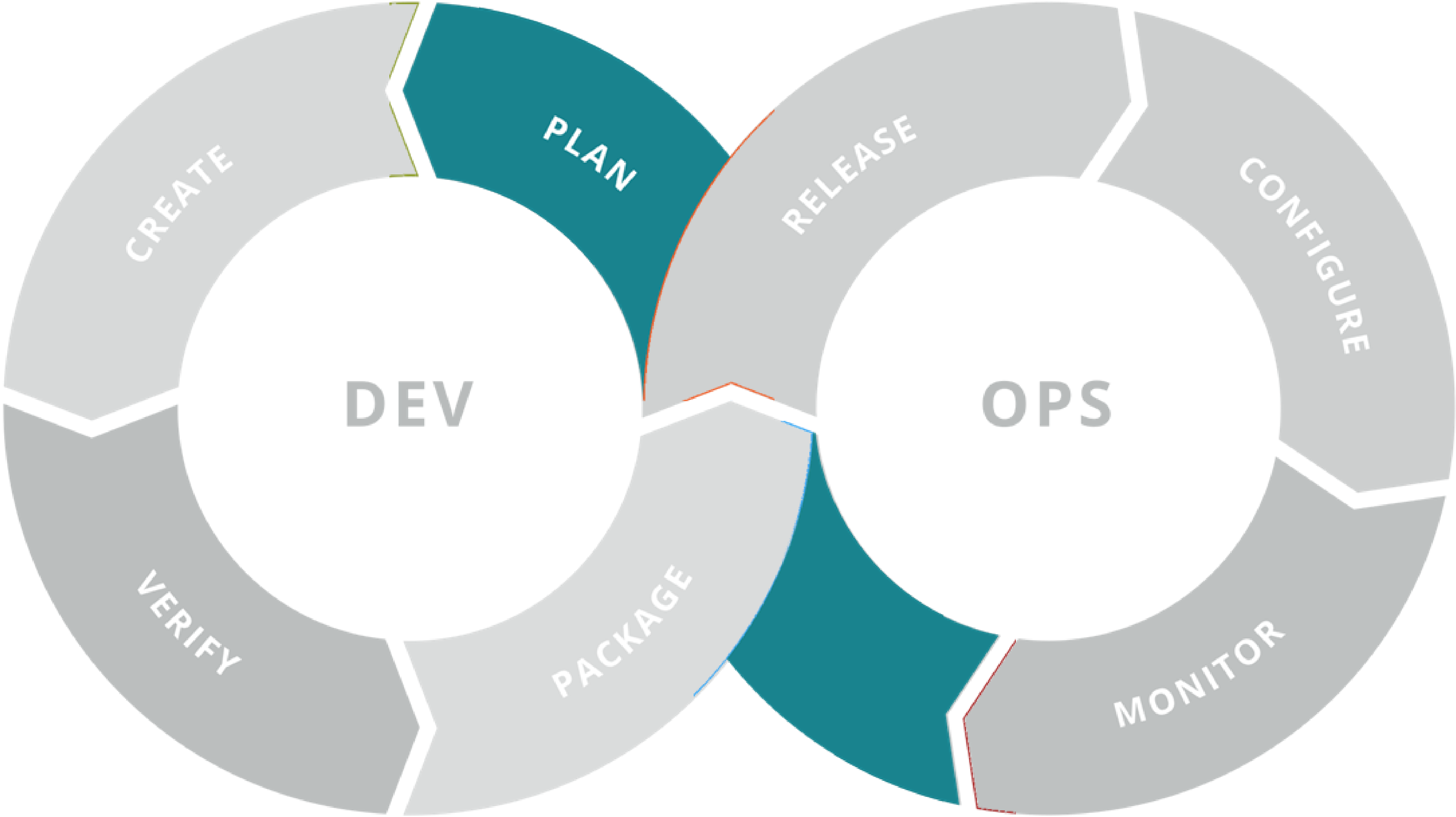
https://en.wikipedia.org/wiki/DevOps_toolchain#/media/File:Devops-toolchain.svg

What’s Next? – Shifting Right



https://en.wikipedia.org/wiki/DevOps_toolchain#/media/File:Devops-toolchain.svg

What’s Next? – Shifting Left?



https://en.wikipedia.org/wiki/DevOps_toolchain#/media/File:Devops-toolchain.svg

What’s Next if I am …

A researcher:

AI will increasingly interact with people – **human subject studies** are key

- Enabling **productive collaboration** with AI agents
- Start with: how do developers currently collaborate & communicate?
- Then: how will that **change**, as more and more SE is done by AI?
- Managing **ownership and responsibilities** in AI-generated/maintained projects

What’s Next if I am …

A researcher:

To make AI effective, we need **new metrics and benchmarks**

- How to evaluate a comment? A design document? An entire PR?
- We need **next-generation benchmarks**, possibly LLM-powered.
- Building **frameworks of performance** to support developers, end-users
- As LLMs enter UX, we need design patterns for agents, test suites for UIs

What’s Next if I am …

A researcher:

To democratize code and AI, we need to **rethink programming**

- Enabling code generation for **8 billion non-programmers**
- **Supporting learning**, debugging, maintenance for end-users
- Bringing together programmers and non-programmers
  - New development environments. Developers as a Service.

What's Next if I am …

A CS student/professional:

Learn the **tools**

Commercial: Copilot, ChatGPT, Bard, Claude, …

Open-Source: InCoder, StarCoder (bidirectional context), CodeGen (strong on Python), (Code)LLaMa (particularly large)

This list **changes constantly**, so important to stay up to date

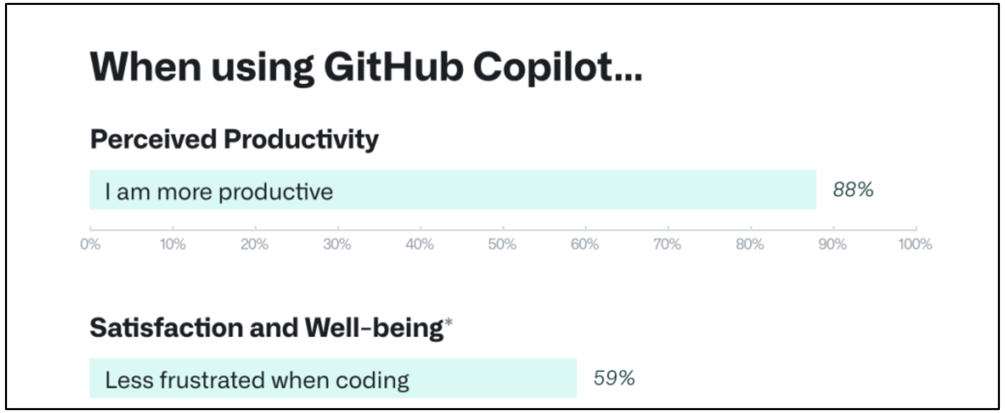
Not a one-way street: **add value** with discernment, planning

What's Next if I am …

A CS student/professional:

Shift the **emphasis** of your work

- Let AI do the **tedious** stuff
- Matplotlib's parameters have ridiculous names that you can't remember? Don't try, use AI
- Need a boilerplate website? Go ask a conversational LLM
- The set of boring things AI does easily is growing fast



When using GitHub Copilot…

Perceived Productivity

I am more productive 88%

Satisfaction and Well-being*

Less frustrated when coding 59%

What’s Next if I am …

A CS student/professional:

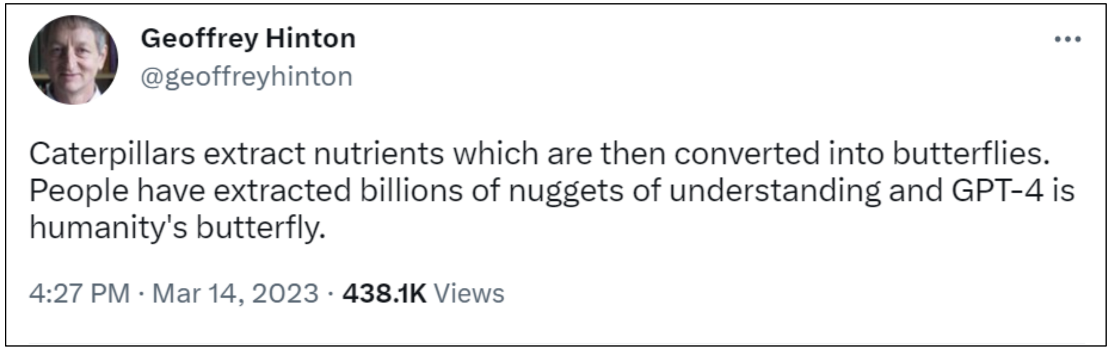
Prepare for AI to do most of the coding

Prioritize **people & design** skills

- AI won’t tell you what to build next, or who uses your products and why

Keep track of **what’s important**

- AI is fueled by our past achievements. It changes what we think is hard & meaningful
- Keeping up with new tools helps you calibrate



Geoffrey Hinton
@geoffreyhinton

Caterpillars extract nutrients which are then converted into butterflies. People have extracted billions of nuggets of understanding and GPT-4 is humanity's butterfly.

4:27 PM · Mar 14, 2023 · **438.1K** Views

What's Next if I am …

A non-programmer:

A near-term future where almost anyone can **almost code**

- Some programming will be like using a screwdriver
- Other times it's more like fixing a car
- Not always obvious which it's going to be

Beyond Code Generation

Towards Next-Generation AI for SE

Vincent J. Hellendoorn

December 3rd, 2023

MAPS @ FSE

vhellendoorn@google.com