AI and Code: Two Projects from GitHub Next

Don Syme (dsyme@github.com) + all of GitHub Next
Principal Researcher, GitHub Next

GitHub Next

Researching the future of
software development

githubnext.com

What is GitHub Next?

What

An applied R&D group attached to GitHub, reports to CEO of GitHub

Mission

Transform the practice of software development

Mode of Operation

***Build**, **Release**, **Learn**, **Co-operate**.*

Who

~15 applied LLM/ML experts (many ex-Copilot), UX experts, CS experts

Why this is the right way to run innovative applied R&D

Operates at the Goldilocks distance!

Released March 23!

Copilot for Docs

March 22, 2023

WAITLIST

How would it feel to have an expert on hand at all times? We built a tool that uses relevant information from a project's documentation to answer questions or explain concepts.

Copilot for Pull Requests

March 22, 2023

WAITLIST

Pull requests are a central part of the GitHub user experience. Copilot for PRs brings the power of Copilot to the PR experience, to help you write better PR descriptions, and to help your team review and merge PRs faster.

Copilot for CLI

March 22, 2023

WAITLIST

Ever having trouble remembering that shell command or this obscure flag? Don't worry: we're building GitHub Copilot assistance right into your terminal

Copilot Voice

March 22, 2023

WAITLIST

Write code without the keyboard. Difficulty typing? Use your voice to code without spelling things out by talking with GitHub Copilot.

Today's Menu

Equipping GPT-4 with Numeric Calculation

Interlude

Towards pervasive summarization in GitHub

Part 1 - Equipping GPT-4 with Numeric Calculation

github.com/githubnext/gpt4-with-calc

caveat

Lots and lots of related work

Equipping for Numeric Calculation

The situation:

- **Any GPT-4 Chat: Context + Question ⇒ Answer**

Equipping for Numeric Calculation

The situation:

- **Any GPT-4 Chat: Context + Question ⇒ Answer**

The problem:

- GPT-4 is **terrible** at numeric calculations
- It's also terrible at numeric comparisons

How bad is this?

It’s bad

Example fails

- Comparing financial documents
- Example problem:

Both companies reported an increase in net sales compared to the same quarter last year, but Gap Inc. had a much **larger** absolute **and** [illegible] increase ($4.04 billion, **up 2%**) than lululemon ($1.9 billion, **up 28%**).

More example problem:

```
Solving for n, we get:

n = ln(2) / ln(1 + 0.3)

Using a [illegible]or, we get:

n ~= 2.53

Therefore, it takes about 2.53 years for an item growing at 30% annually to double.
```

The Hope

The observation:

👎GPT-4 is **terrible** at **numeric calculations**

👍GPT-4 is **good** at writing **numeric calculation code**

The Hope made Real

Without numeric calculation equipping:

1. **GPT-4**: Question --> Answer

With numeric calculation equipping:

1. **GPT-4**: Question --> Numeric calculation code
2. **Evaluator**: Numeric calculation code --> Numeric calculation answers
3. **GPT-4**: Question + Numeric calculation code + Numeric calculation answers --> Answer

My opinion: this two-phase technique should be incorporated into **every** GPT-4 request that involves numbers unless explicitly known to be calculation-free. Anything else is

Example question

This is an extract from the Microsoft annual report for 2022:

Microsoft repurchased the following shares of common stock under the share repurchase programs:

(In millions)	Shares	Amount	Shares	Amount	Shares	Amount
Year Ended June 30,		2022		2021		2020
First Quarter	21	$6,200	25	$5,270	29	$4,000
Second Quarter	20	6,233	27	5,750	32	4,600
Third Quarter	26	7,800	25	5,750	37	6,000
Fourth Quarter	28	7,800	24	6,200	28	5,088

What is the overall average and standard deviation of shares repurchased per quarter over these three years in dollars?

Example fail

...lots of working ...

Therefore, the overall average and standard deviation of shares repurchased per quarter over these three years in dollars ar[illegible] million dollars and 1[illegible] million dollars, respectively.

Example prompt

```
# Guidance

Do not answer the question. Instead, your task is to write some numeric
calculation and comparisons relevant to answering the question.

After the question write a code block with up to three sections containing
content relevant to answering the question.

In the "Definitions" section define a label for each number in the
original question like `car_count` or `speed_of_car_in_km_per_hour`.
* Every label name should include the unit of measure if known.
* This section should be valid Python and can include valid Python
  single-dimensional arrays.
* Do not use or create multi-dimensional arrays.
* Give each label a unit of measure in a comment after each definition.
* Document the meaning of each definition in the comment.
* If the unit of measure is unknown use "unknown".
* Omit this section if there are no numbers in the question.

In the "Calculations" section define additional relevant labels using
```

Example relevant calculation code

```
# Definitions
shares_2022 = [21, 20, 26, 28] # millions of shares
amount_2022 = [6200, 6233, 7800, 7800] # millions of dollars
shares_2021 = [25, 27, 25, 24] # millions of shares
amount_2021 = [5270, 5750, 5750, 6200] # millions of dollars
shares_2020 = [29, 32, 37, 28] # millions of shares
amount_2020 = [4000, 4600, 6000, 5088] # millions of dollars

# Calculations
total_shares = shares_2022 + shares_2021 + shares_2020 # millions of shares
total_amount = amount_2022 + amount_2021 + amount_2020 # millions of dollars
average_amount = np.mean(total_amount) # millions of dollars
std_amount = np.std(total_amount) # millions of dollars
```

Example relevant results

```
total_shares [21, 20, 26, 28, 25, 27, 25, 24, 29, 32, 37, 28]
total_amount [6200, 6233, 7800, 7800, 5270, 5750, 5750, 6200, 4000, 4600, 6000, 5088]
more_shares_2022 False
more_amount_2022 True
average_amount 5890.916666666667
std_amount 1077.077175997874
```

Add code + results to the prompt, before the question

Example answer

The overall average and standard deviation of shares repurchased per quarter over these three years in dollars are:

- Average: $5,890.92 million
- Standard deviation: $1,077.08 million

...lots of working ...

Therefore, the overall average and standard deviation of shares repurchased per quarter over these three years in dollars are 5890.92 million dollars and 1077.08 million dollars, respectively.

Effective?

Raw calculation (decimals, integers, math functions):

- without equip: **50% mistake rate** (77/153), many extreme mistakes
- with equip: **0% mistake rate** (153/153)

Financial calculations (mortgage, rate of return, WACC):

- without equip: **57% mistake rate**, 4/14
- with equip: **14% mistake rate** 12/14

Tabular data calculations:

- without equip: **50% mistake rate** 4/8
- with equip: **0% mistake rate** 8/8

Unit conversion calculations problems

- without equip: **36% mistake rate** 7/11
- with equip: **0% mistake rate** 11/11

Date calculations:

- without equip: **30% mistake rate** 7/10
- with equip: **20% mistake rate** 8/10

Word puzzles:

- Without equip: **11% mistake rate** (253 failures 2050/2303)
- With equip: **6.2% mistake rate** (145 failures 2158/2303)

Some mantras

Calculation is hard (for AI). Writing calculation code is (relatively) easy.

Computation is hard (for AI). Writing code is (relatively) easy.

("Hey AI bro, you need that code after all?")

Bonus topics

- It’s not just **calc**, it’s **comparisons** too
- **Units of measure** can be extracted and “tracked”
- Self-**checks**
- **Commenting** your calculations matters

Numeric calculation is the “hello world” of trustworthy systems

Numbers are always “rational”

INTERLUDE:
CAVEAT EMPTOR!

Welcome to the Future

The integration of **Chat, Data** and **Programming** is coming very fast. Much faster than I expected even a month ago.

Important: chat is not just one thing:

- Chat for Entertainment
- Chat for Search
- **Chat for Analytical Question Answering**
- **Chat for Technical Creation**
- **Chat for Tax Returns**

“Code Interpreter” ChatGPT AddIn

Code interpreter Alpha

An experimental ChatGPT model that can use

We p… g Python
envi… neral disk
evalu… is alive f
uppe… nt calls c
uplo… sation wo

Ethan Mollick @emollick

This was kind of delightful: I uploaded a CSV file of every lighthouse location in the US.

"ChatGPT Code Interpreter: Create a gif of a map of the lighthouse locations, where the map is very dark but each lighthouse twinkles." A couple seconds later...

GIF

Boost Your Data Workflow with Noteable and ChatGPT

ChatGPT Plugin for Jupyter Notebooks

Get Started

JW

a brick weighs 30% of a brick plus five GPUs there are 100 bricks and 200 GPUs in a metric ton how much does a brick weigh?

JW

a brick weighs 30% of a brick plus five GPUs there are 100 bricks and 200 GPUs in a metric ton how much does a brick weigh?

Let's call the weight of a brick "B" and the weight of a GPU "G". We can set up an equation based on the information given:

B = 0.3 * B + 5 * G

We can also set up another equation based on the total weight of 100 bricks and 200 GPUs being equal to 1 metric ton (1000 kilograms):

100 * B + 200 * G = 1000

We can solve this system of equations to find the weight of a brick (B) and the weight of a GPU (G).

Finished calculating

The weight of a brick is 7.8125 kg.

ode interpreter plugin performing logic calculations

```
from sympy import symbols, Eq, solve

B, G = symbols('B G')
eq1 = Eq(B, 0.3 * B + 5 * G)
eq2 = Eq(100 * B + 200 * G, 1000)
solution = solve((eq1, eq2), (B, G))
solution[B]
```

```
# Result
7.81250000000000
```

What's Truly Needed for Trustworthy Chat?

The industry is crying out for AI that can be ***trusted***, can be ***explained***, can generate ***trustworthy***, ***recognisable***, ***analyzable*** artifacts like ***spreadsheets*** and ***notebooks***.

By generating calculation code, we have a stepping stone to this.

The Good News

This category of technique can be applied to **any language**, **computational logic**, **DSL, query language, library**, **service** or **computational capability**.

The outputs can be **answers**, **code**, **notebooks**, **proofs**, **spreadsheets**, **projects**, **tests**, **documentation. And tax returns.**

The Devil is in the Detail

Every step of an LLM can introduce hallucination.

Example: **hallucinated numbers in calculation code**

```
# Calculations
gap_gross_profit = gap_net_sales * gap_gross_margin # billion USD
lulu_net_revenue_growth = (lulu_net_revenue - 1.5) / 1.5 # fraction
lulu_store_revenue_growth = (lulu_store_revenue - 0.488) / 0.488 # fraction
lulu_ecommerce_revenue_growth = (lulu_ecommerce_revenue - 0.404) / 0.404 # fraction
gap_inventory_growth = (gap_inventory - 2.72) / 2.72 # fraction
lulu_inventory_growth = (lulu_inventory - 0.9) / 0.9 # fraction
gap_capital_expenditures_growth = (gap_capital_expenditures - 308) / 308 # fraction
lulu_capital_expenditures_growth = (lulu_capital_expenditures - 122.5) / 122.5 # fraction
gap_dividend_yield = gap_dividend * 4 / gap_diluted_eps # fraction
lulu_dividend_yield = 0 # fraction
gap_share_repurchases_yield = gap_share_repurchases / gap_net_income # fraction
lulu_share_repurchases_yield = lulu_share_repurchases / lulu_net_income # fraction
```

Questions for Today

- Hardened, reliable, trustworthy calculation/reasoning systems are **utterly critical**
  - Proof, evidence, reasoning, explanation, audit, provenance and blame are critical in trust domains
- In this world, code is much more ephemeral
  - What is “correct” code?
  - What is “evidence” that the chosen code or query is correct?
  - What hard engines help give establish trust and confidence?
  - What assumptions have been made?

Part 2 - Summaries in GitHub

githubnext.com/projects/copilot-for-pull-requests

Summarization: Rationale

We are headed to a world where **most text associated with repos** is **emergent** and **automatic.**

Content ⇒ Search **Content ⇒ Text**

Uses:

- Abstracts, keywords, descriptions = for comprehension
- Table of contents = for navigation
- Documentation, walkthroughs = for explanation

Use cases:

- Tool-tips Infopanel content
- Seeding human-authored descriptions Search

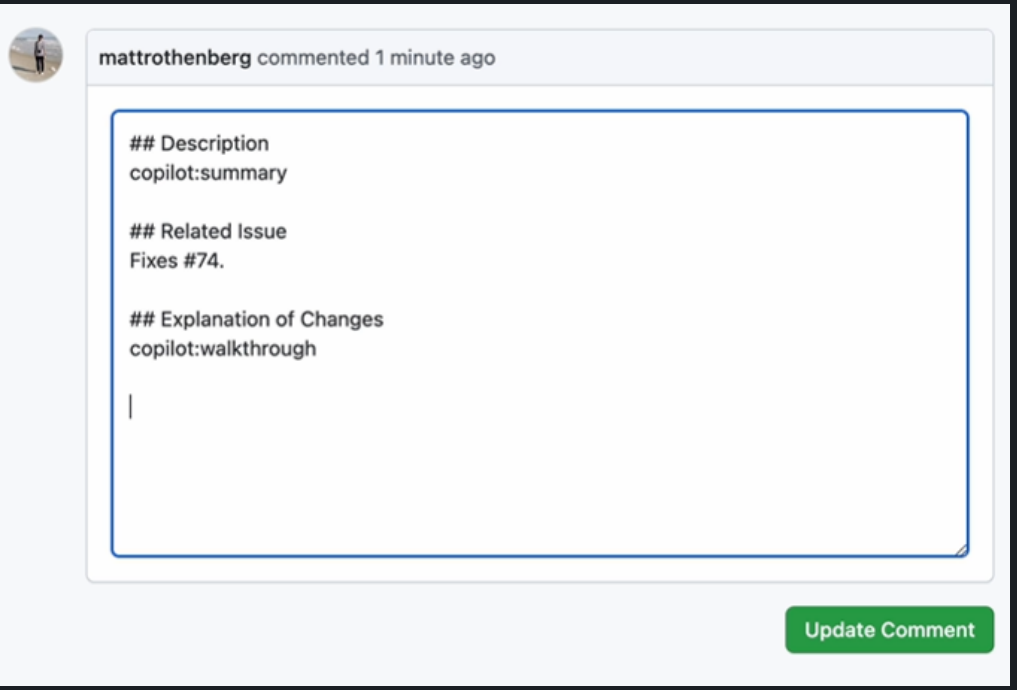
Esp. when **unfamiliar** with content, **pressed for time**, or **small-factor devices**

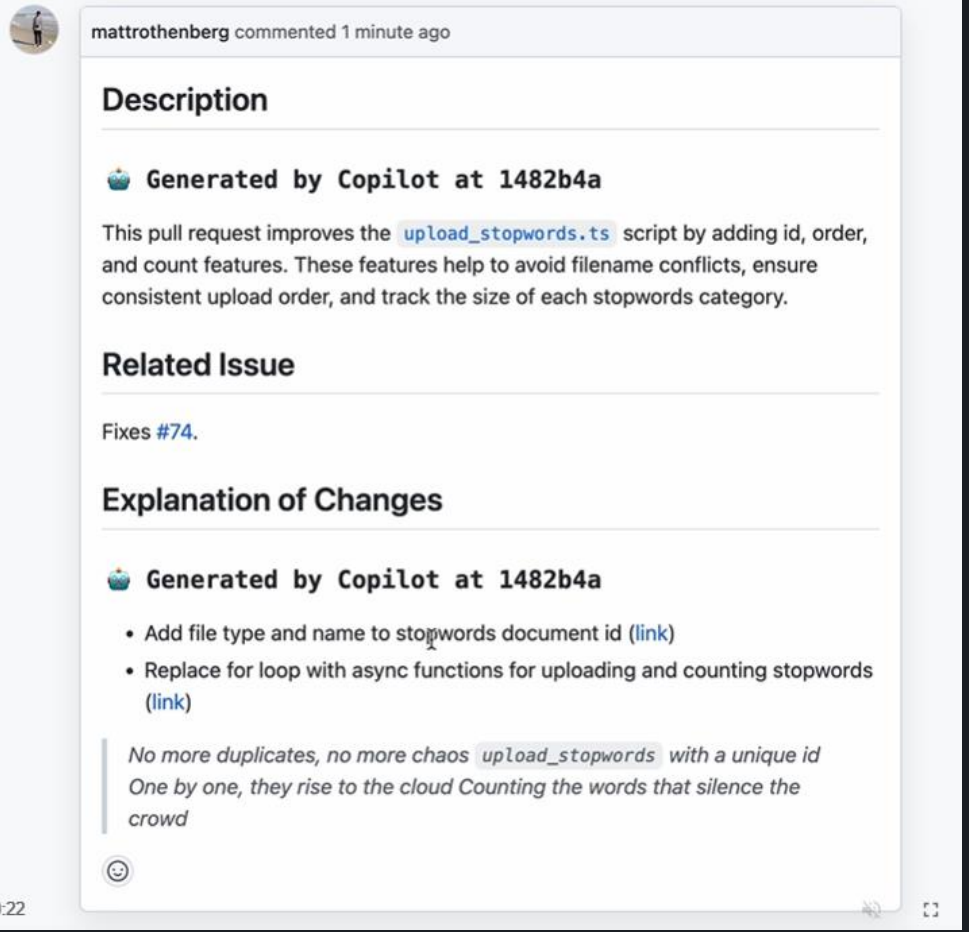
Example – PR Summary

./prbot describe https://github.com/githubnext/prbot/pull/565

▼ Summary

This pull request enhances the bot's ability to describe code changes, updates the test scores for the describe task, and simplifies the poetry generation code. It modifies `src/bot/bot-describe.ts` to improve the description quality and readability, updates `test/scores/boosted/describe-score.md` to report the latest results, and removes an unused case from `src/engine/poems.ts` to improve code clarity.


mattrothenberg commented 1 minute ago
Description
copilot:summary
Related Issue
Fixes #74.
Explanation of Changes
copilot:walkthrough
Update Comment



mattrothenberg commented 1 minute ago

Description

Generated by Copilot at 1482b4a

This pull request improves the `upload_stopwords.ts` script by adding id, order, and count features. These features help to avoid filename conflicts, ensure consistent upload order, and track the size of each stopwords category.

Related Issue

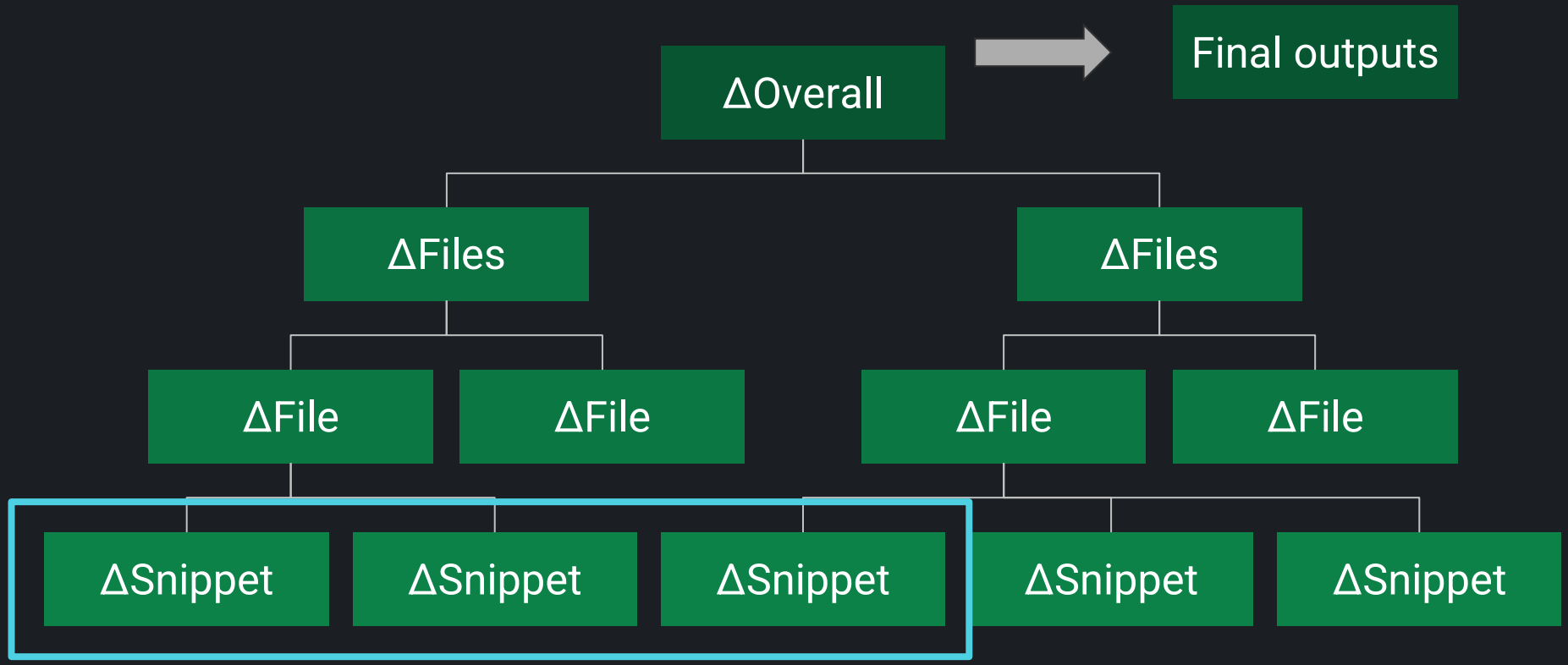
Fixes #74.

Explanation of Changes

Generated by Copilot at 1482b4a

- Add file type and name to stopwords document id (link)
- Replace for loop with async functions for uploading and counting stopwords (link)

> *No more duplicates, no more chaos `upload_stopwords` with a unique id*
> *One by one, they rise to the cloud Counting the words that silence the crowd*


ΔOverall
Final outputs
ΔFiles
ΔFiles
ΔFile
ΔFile
ΔFile
ΔFile
ΔSnippet
ΔSnippet
ΔSnippet
ΔSnippet
ΔSnippet

Sample Prompts

- △Snippet(s) → summary
- △Repository → walkthrough

What about Hallucinations?

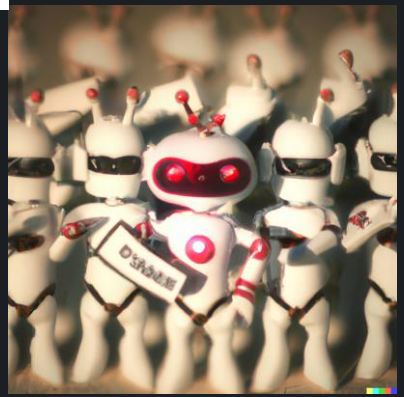
Hallucination Reduction in Summarization

- Label all changes with “locators” for evidence
  - Requiring locators forces LLM to “ground” its text
  - Any change description without evidence is **thrown away**
  - Locators provide hyperlinks back to source
- Small solitary changes hallucinate the most
  - Pack changes together
  - Don’t ask for long descriptions of small changes

A bit of fun - AI Poetry for PRs

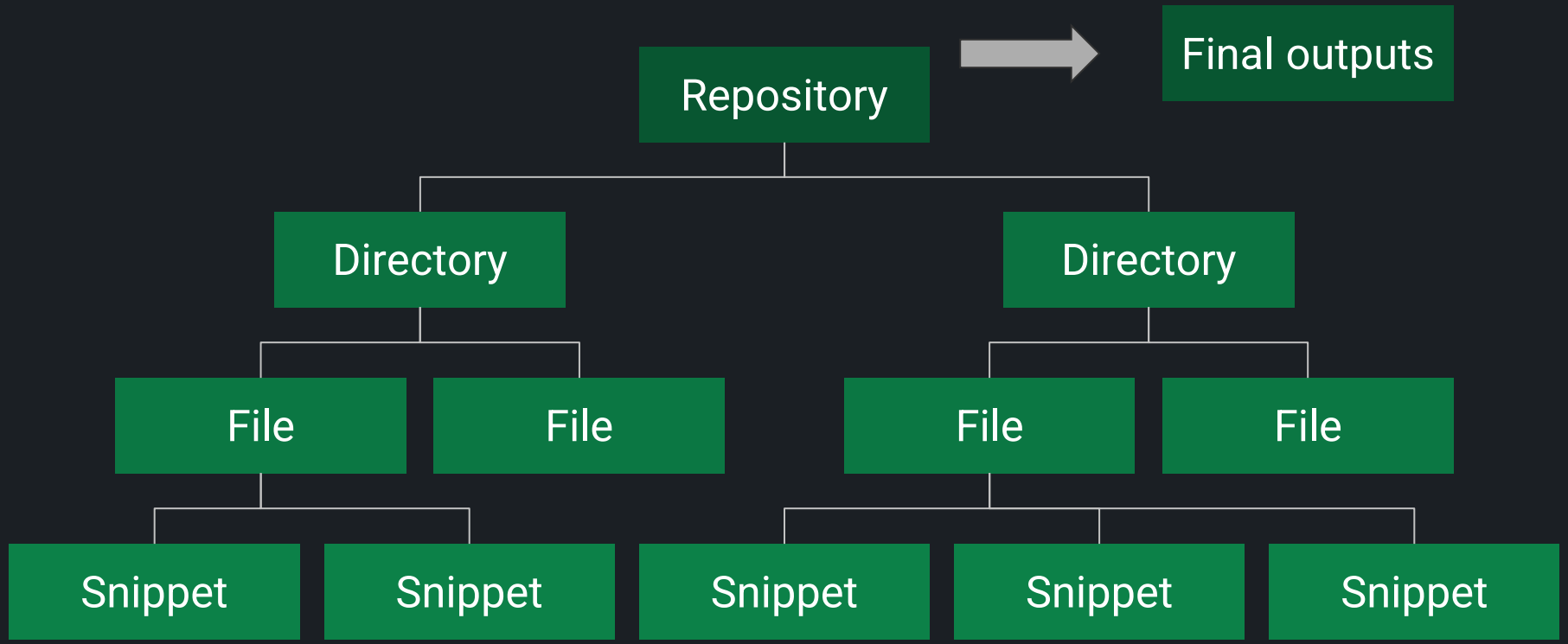
Oh, we're the crew who code the bot, and we're the best you'll ever see
We review the changes in the files, with a `fileIndex` *for each*
We heave and ho and pull the rope, and make the locators neat
We're the crew who code the bot, and we're the pride of the fleet

Command line **`prbot-engine`**
Added for automation ease
Springtime coding bliss **`#!`**



We unleash the prbot, the beast of AI
We measure its power, its skill and its price
We refine its language, its summaries and scores
We challenge the baseline, the human and the norms

Summarizations: From PRs to Entire Repos


Repository
Final outputs
Directory
Directory
File
File
File
File
Snippet
Snippet
Snippet
Snippet
Snippet

Hierarchical summaries of entire repositories

Overall summary of 1M+ LOC code in github/github “app” directory.

Every chunk, file, directory, issue, discussion gets a summary too.

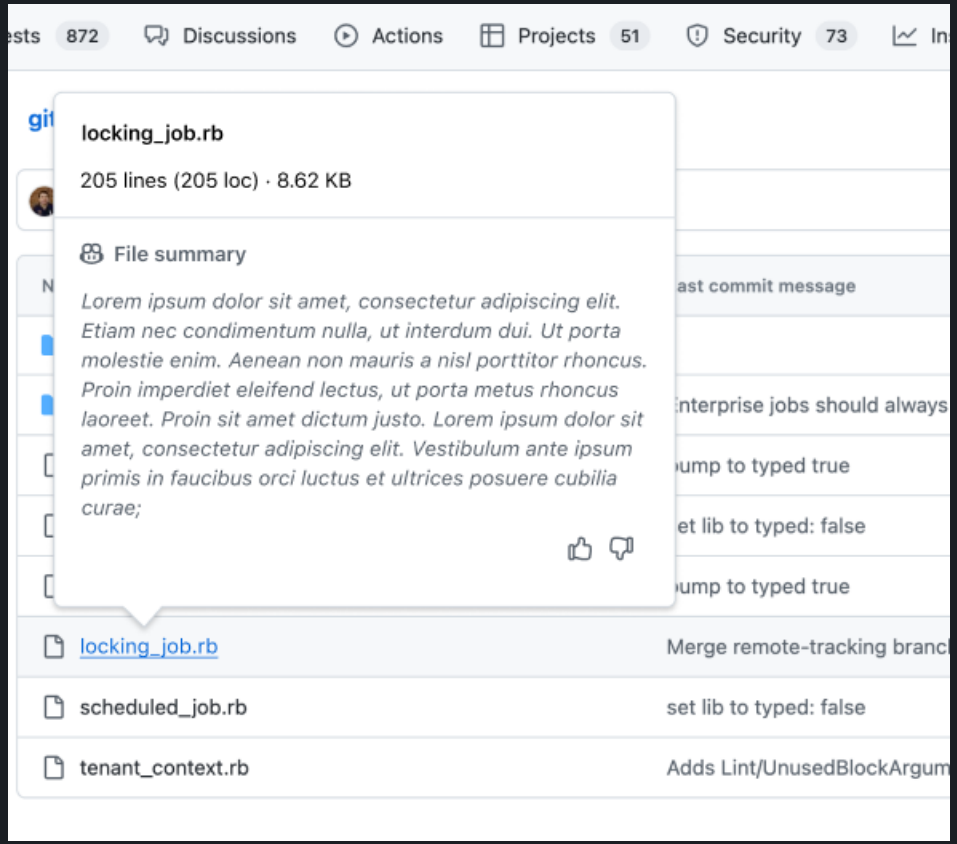
The directory `app/` contains files and subdirectories for the GitHub web application, which allows users to create, manage, and collaborate on software projects.

- `app/api/`
  - The GitHub API, which provides programmatic access to GitHub features and data.
  - Includes core API files, serializers, and subdirectories for various features such as organization, repository, code scanning, team, search, runtime, staff tools, enterprise, codespaces, deployments, and environment.
- `app/components/`
  - View components for various features and pages of the GitHub web application.
  - Uses the view_component library and the sorbet type checker.
  - Includes components for account verifications, GitHub Actions, GitHub Advanced Security, security advisories, animation, apps, beta features, billing, blobs, branch, businesses, checks, closables, code navigation, code scanning, codespaces, command palette, comments, commits, compare, compliance, conditional access, conduit, config as code, context switcher, copilot, dashboard, dependabot, dependency graph and review, devtools, diffs, discussions, environments, events, explore, feed, feed posts, files, fragments, flipper features, gists, and hovercards.
- `app/controllers/`

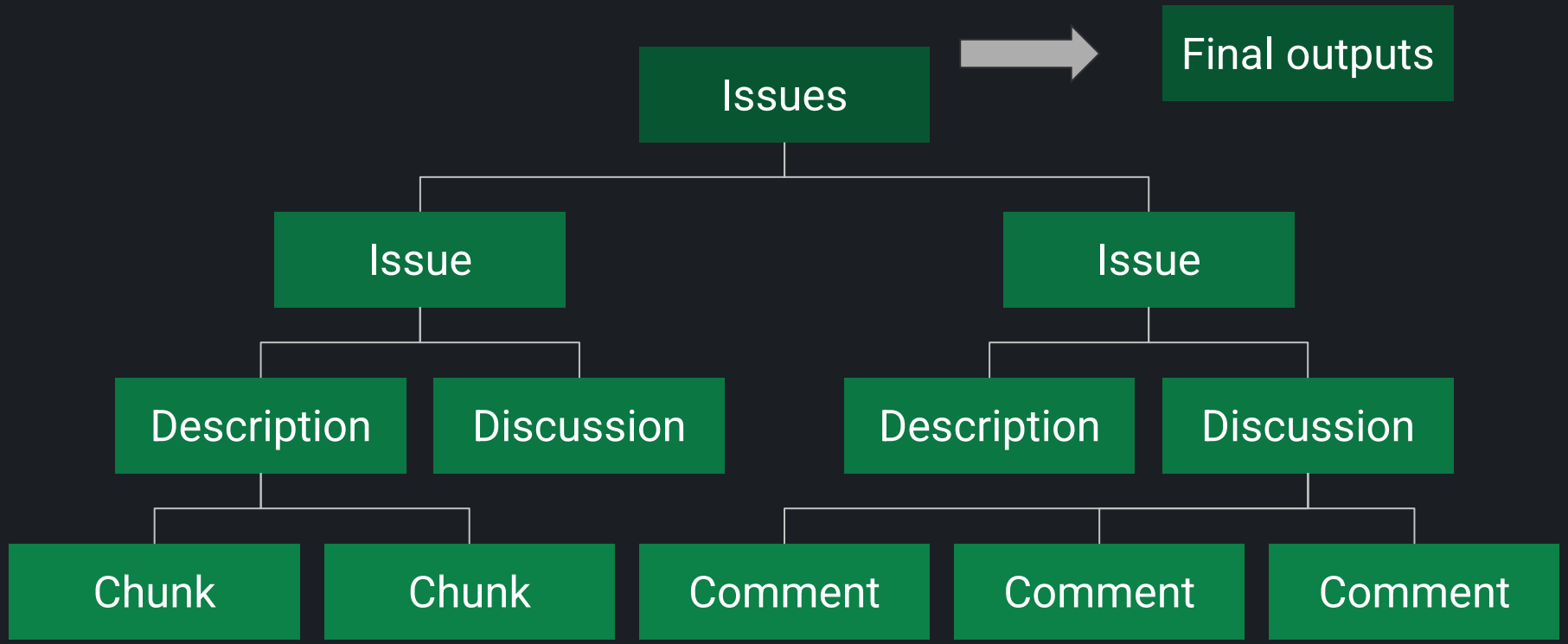
Surfacing in the UX

scorer	*Scripts and data for evaluating prbot summaries and walkthroughs.*	2 weeks ago
src	*Source code for prbot, a GitHub bot using OpenAI models.*	3 hours ago
test	*Tests, baselines, and scores for the prbot tool.*	4 hours ago
.dockerignore	*Files and directories to exclude from Docker build context.*	6 months ago
.env.example	*Environment variables for prbot, including credentials and proxy URL.*	4 months ago
.eslintignore	*Files and directories to exclude from ESLint.*	4 months ago
.eslintrc	*ESLint configuration for TypeScript files with plugins and rules.*	4 months ago

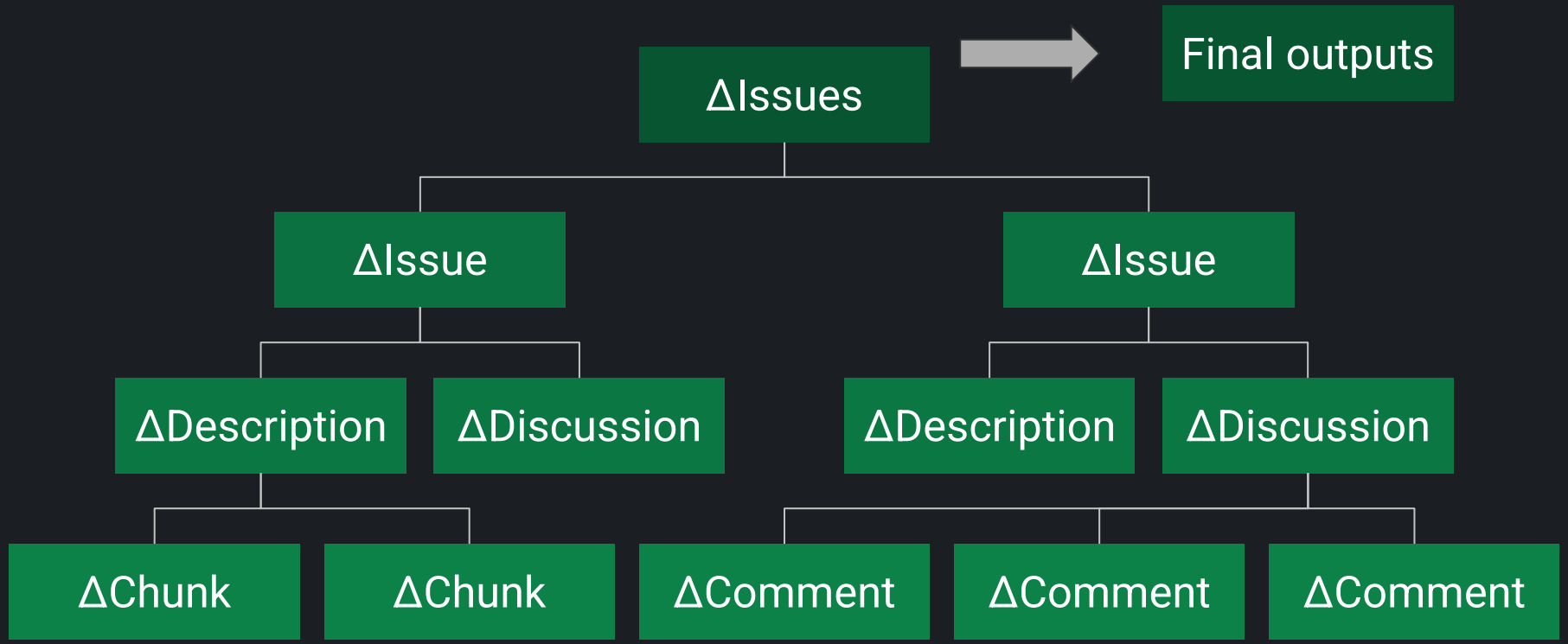
Surfacing in the UX



“Summarize all issues in this repo”



“Summarize discussions in this repo over last week”



Thank you!

dsyme@github.com

From Code to Issues and more

Determining Hallucination Rates

- This is hard
- Subject Matter Experts must assess
- Ran a “Hallucination Hunter” game internally to incentive experts to find hallucinations in descriptions of their own code
  - Very few in final summaries

RLA

- RLA = **Required Logical Accuracy**
- Same outputs have different RLA for different tasks!
  - Code Review = very high RLA
  - “A Quick Overview” for the Newcomer = medium RLA
  - Poem = low RLA
- UX really matters, can reduce RLA

Summarize all the things

Code ⇒ Summary (“Explain this code” in Copilot VS Code)

PR ⇒ Summary (Copilot for PRs)

File ⇒ Summary

Directory ⇒ Summary

Issue ⇒ Summary

Discussion ⇒ Summary

Repo => Summary

(multiple AI calls needed == hierarchical summarization)