

LLMs are not the product

Integrations beyond chatbots



Jason Etcovitch
Staff Software Engineer @GitHub

Ask the chatbot something...

Jason Etcovitch

Staff Software Engineer @GitHub

Copilot Platform-y things

@jasonetc0 everywhere



I am not a data scientist

I am not a machine learning person

I am not a AI expert

I am a regular ol' **software engineer**

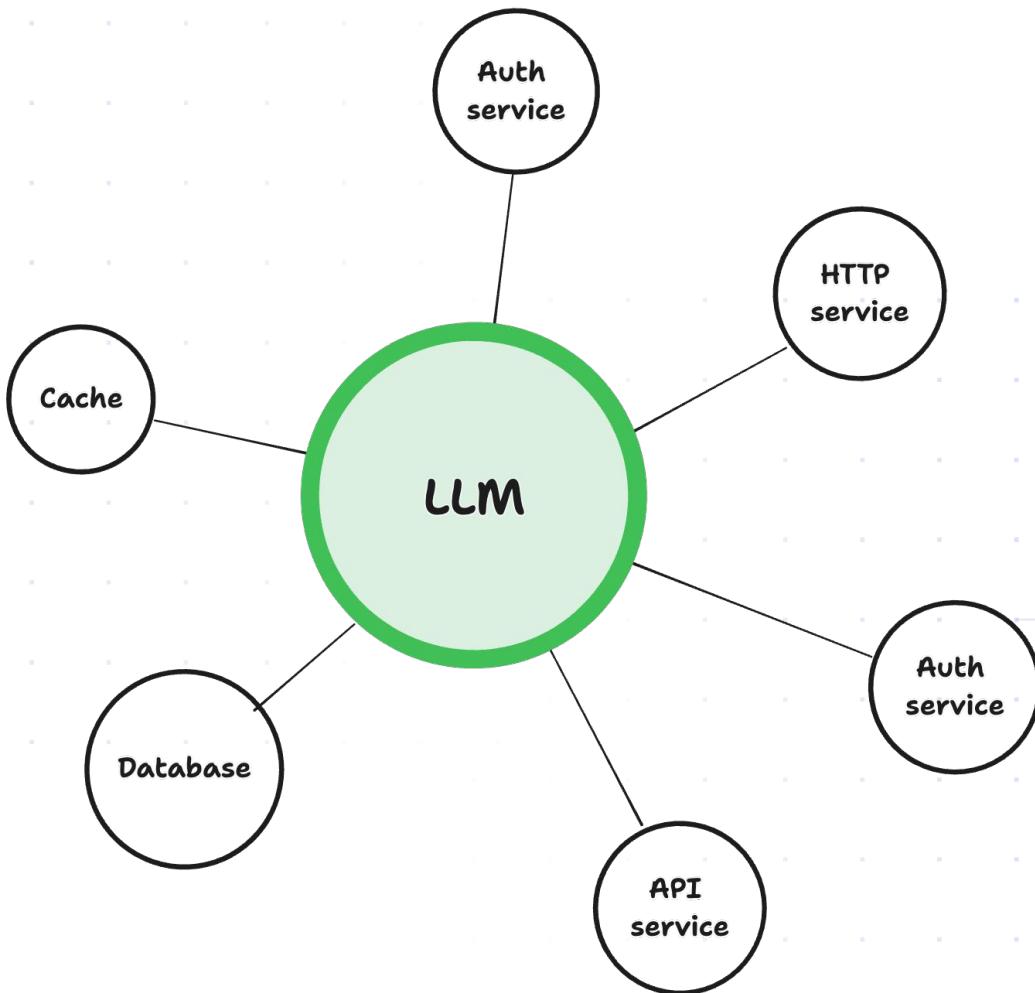
“

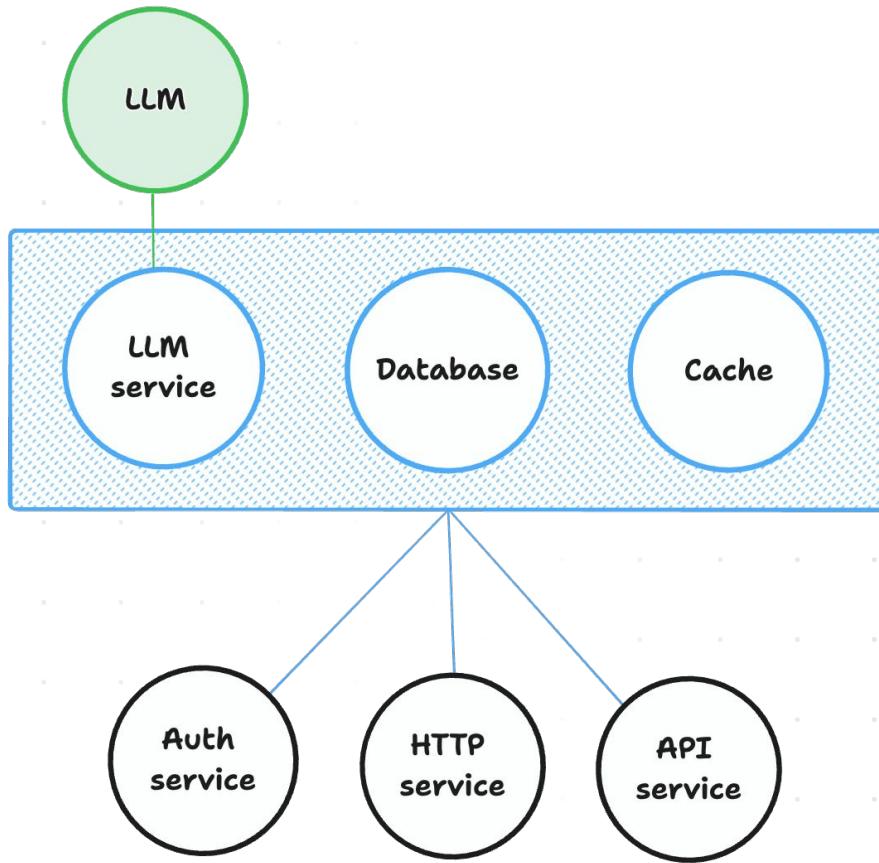
Eventually, AI will be another tool in the product engineering toolbelt that lives alongside well-established concepts like databases and caches.



Anonymous GitHubber

A couple of months ago





“ ”

LLMs are **not magic.**

“ ”

LLMs won't save you from a bad UX.

“ ”

Chat is the least structured interface for *using* an AI.

“

Chat experiences are limiting in how vague they are.



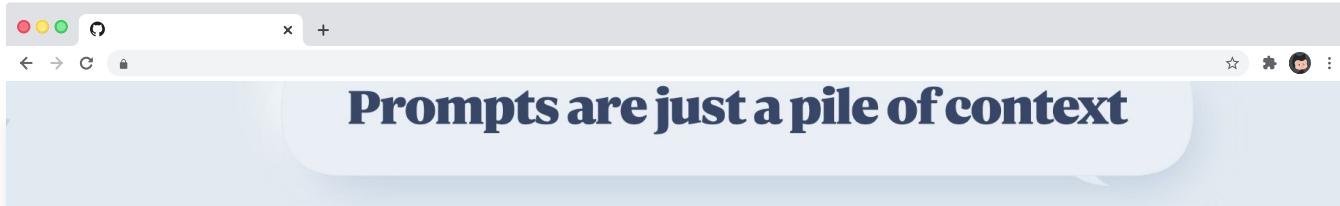
Me
Right now

ChatGPT

A screenshot of the ChatGPT web interface. The window title is "ChatGPT". The sidebar on the left shows a list of previous conversations: "Today" includes "Questions about 4th of July", "Capital: Ottawa"; "Previous 7 Days" includes "cURL with Array Header"; "Previous 30 Days" includes "User Request: Testing Assista"; "June" includes "Square root of 7.". The main chat area shows a user message "What is the square root of 7?" followed by a GPT response "The square root of 7 is approximately 2.6457513110645907." At the bottom, there is a "Regenerate response" button and a "Send a message" input field. The footer includes links for "Upgrade to Plus" and "Jason Etcovitch", and a note: "Free Research Preview. ChatGPT may produce inaccurate information about people, places, or facts. ChatGPT May 24 Version".

Why Chatbots Are Not the Future

<https://wattenberger.com/thoughts/boo-chatbots>



LLMs make it too easy: we send them text and they send back text. The easy solution is to slap a shallow wrapper on top and call it a day. But pretty soon, we're going to get sick of typing all the time. If you think about it, **everything you put in a prompt is a piece of context**.

Let's look at a simple example from [Awesome ChatGPT Prompts](#):

Who are you? I want you to act as a dream interpreter. I will give you descriptions of my dreams, and you will provide interpretations based on the symbols and themes present in the dream. Do not provide personal opinions or assumptions about the dreamer. Provide only factual interpretations based on the information given. My first dream is about being chased by a giant

How should you respond? How should you not respond? What type of information do I want? How should we start?

“ ”

The interface looks the same as a Google search box, a login form, and a credit card field.



@wattenberger

<https://wattenberger.com/thoughts/boo-chatbots>

“ ”

Good tools make it clear how they should be used.

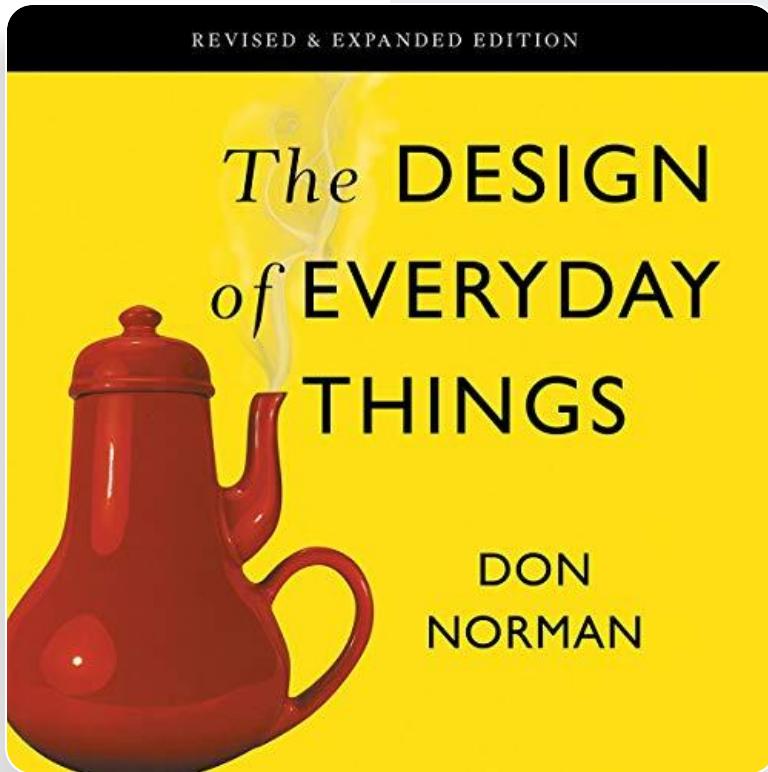


@wattenberger

<https://wattenberger.com/thoughts/boo-chatbots>

Norman doors

The Design of Everyday Things -
Don Norman







Ask the chatbot something...

GitHub Copilot X

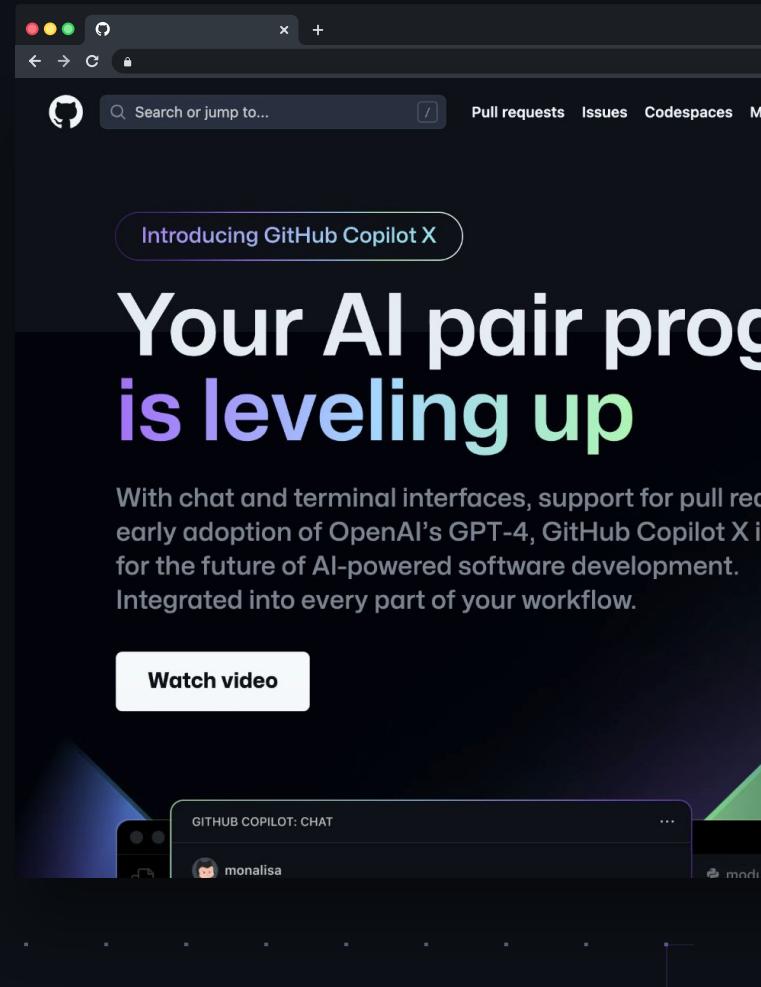
<https://github.com/features/preview/copilot-x>

GitHub Copilot Chat

Context aware conversations with
your copilot.

AI-generated PR descriptions.

GitHub Copilot keeps track of your
work, suggests descriptions...



GitHub Copilot Chat

The screenshot shows a GitHub Copilot Chat window integrated into a dark-themed code editor. The window title is "GITHUB COPILOT: CHAT". Inside, monalisa asks for unit test functions for the selected code. GitHub Copilot provides a template for a Python test class named `TestParseExpenses`. The code includes three test methods: `test_parse_expenses_with_valid_input`, `test_parse_expenses_with_empty_input`, and `test_parse_expenses_with_comments`. It also includes a main block with `unittest.main()`. The right side of the image shows the corresponding Python file `module.py` with the generated code. The code defines a function `parse_expenses` that parses a string of expenses into a list of tuples. It uses the `datetime` module to parse dates and `float` to parse amounts. A sample input string and its expected output are shown.

```
import unittest
import datetime

class TestParseExpenses(unittest.TestCase):
    def test_parse_expenses_with_valid_input(self):
        expenses_string = '2023-01-02 -34.01 USD\n2023-01-03 2.59 DKK
        expected_output = [(datetime.datetime(2023, 1, 2),
        self.assertEqual(parse_expenses(expenses_string), expected_output)

    def test_parse_expenses_with_empty_input(self):
        expenses_string = ''
        expected_output = []
        self.assertEqual(parse_expenses(expenses_string), expected_output)

    def test_parse_expenses_with_comments(self):
        expenses_string = '# This is a comment\n2023-01-02 -34.01 USD
        expected_output = [(datetime.datetime(2023, 1, 2),
        self.assertEqual(parse_expenses(expenses_string), expected_output)

if __name__ == '__main__':
    unittest.main()

def parse_expenses(expenses_string):
    """
    Parse the list of expenses and return the list of triples (date, amount, currency).
    Ignore lines starting with #.
    Parse the date using datetime.
    Example expenses_string:
        2023-01-02 -34.01 USD
        2023-01-03 2.59 DKK
        2023-01-03 -2.72 EUR
    """
    expenses = []
    for line in expenses_string.splitlines():
        if line.startswith("#"):
            continue
        date, value, currency = line.split(" ")
        expenses.append((datetime.datetime.strptime(date, "%Y-%m-%d"),
                        float(value),
                        currency))
    return expenses

expenses_data = '''2023-01-02 -34.01 USD
2023-01-03 2.59 DKK
2023-01-03 -2.72 EUR'''
```

The code assumes that the `datetime` module is imported. The test cases cover different scenarios such as valid input, empty input, input with comments, invalid date format, and invalid value format.

Analyze and debug

Ask a question or type '/' for topics

Ln 17, Col 3 Spaces: 2 UTF-8 LF () TypeScript

Analyze and debug

“Icebreakers” - more entry points for chat capabilities!

The screenshot shows a dark-themed AI interface. On the left, there's a sidebar with icons for user profile, settings, and a message input field labeled "main". The main area displays a Python script with code for testing expense parsing:

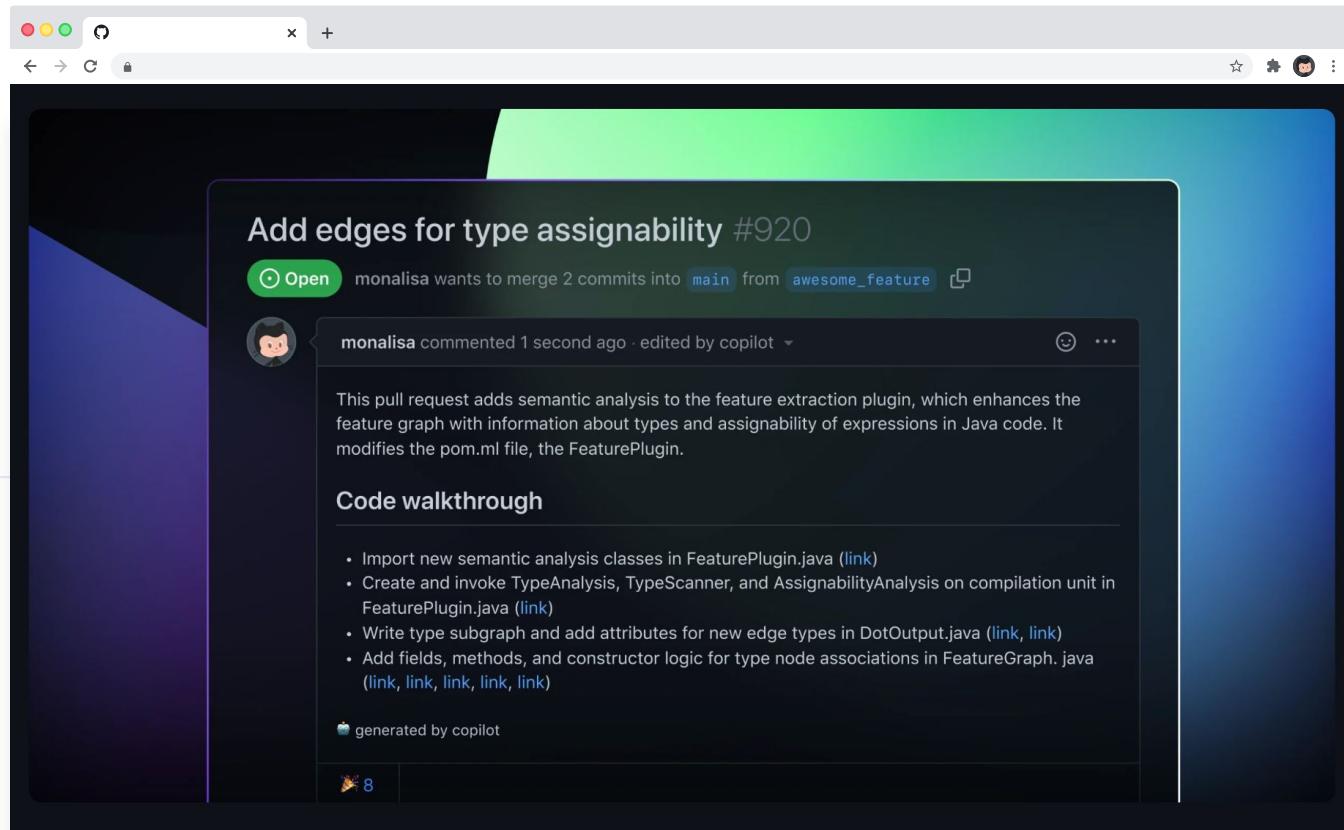
```
def test_parse_expenses_with_empty_input():
    expenses_string = ''
    expected_output = []
    self.assertEqual(parse_expenses(expenses_string), expected_output)

def test_parse_expenses_with_comments(self):
    expenses_string = '# This is a comment\n2023-01-01 100.00\n'
    expected_output = [(datetime.datetime(2023, 1, 1), 100.0)]
    self.assertEqual(parse_expenses(expenses_string), expected_output)

if __name__ == '__main__':
    unittest.main()
```

Below the code, a note states: "The code assumes that the datetime module is imported. The test cases cover different scenarios such as valid input, entries with comments, invalid date format, and invalid value". A blue button labeled "Analyze and debug" is present, along with a message input field containing the placeholder "Ask a question or type '/' for topics".

AI-generated PR descriptions



“

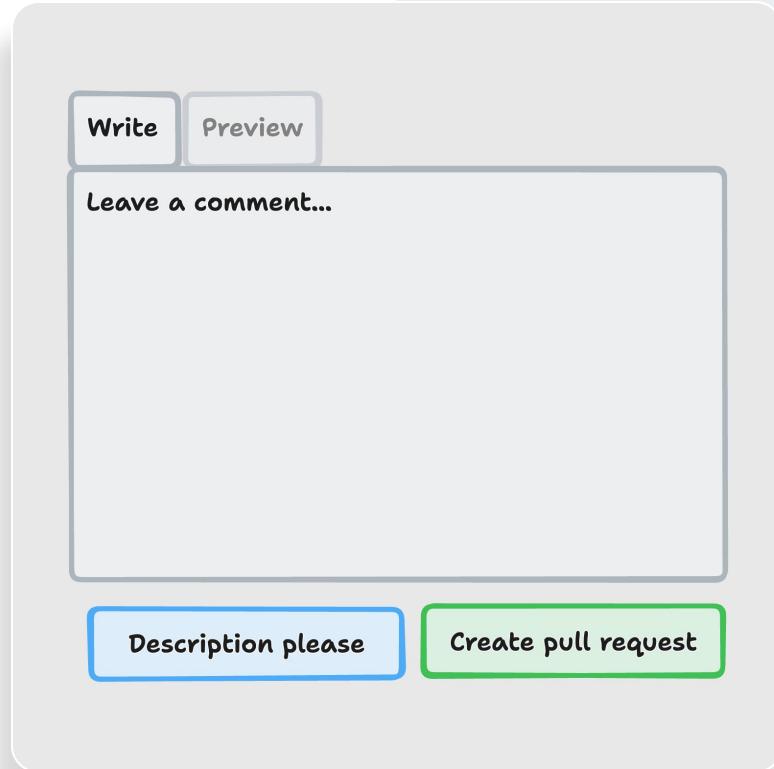
Not all chatbots are bad!
They just need a little guidance.

A chatbot, probably

Generate

Start with the LLM's most generic attempt using context like the diff, repo metadata...

(this is imaginary and not real)



Refine

User provides directional input,
plus initial context, to get closer
to what the user really wants.

(this is imaginary and not real)

This pull request adds semantic analysis to the feature extraction plugin, which enhances the feature graph with information about types and assignability of expressions in Java code. It modifies the pom.xml file, the FeaturePlugin.

Code walkthrough

- * Import new semantic analysis classes in FeaturePlugin.java ([link](#))
- * Create and invoke TypeAnalysis, TypeScanner, and...

Now with more detail on the new 'TypeAnalysis' class please!

Generate++

The chatbot returns a new response, one that is tuned to what the user is aiming for.

(this is imaginary and not real)

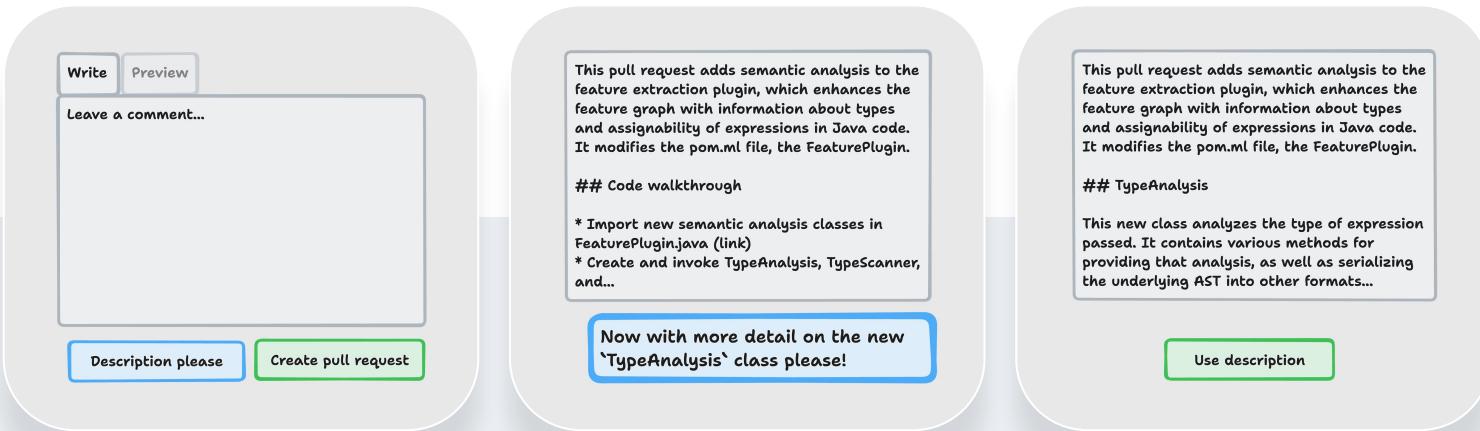
This pull request adds semantic analysis to the feature extraction plugin, which enhances the feature graph with information about types and assignability of expressions in Java code. It modifies the pom.xml file, the FeaturePlugin.

TypeAnalysis

This new class analyzes the type of expression passed. It contains various methods for providing that analysis, as well as serializing the underlying AST into other formats...

Use description

Imaginary PR description flow



Generate

Start with the LLM's most generic attempt

Refine

Take directional input, plus initial context

Generate again

Go again until you're happy with the result

Hallucinations

The issue is related to hallucinations, which occur when AI, specifically a large language model (LLM) such as ChatGPT, generates factually incorrect or nonsensical information that may look plausible.

Jun 7, 2023



SecurityWeek

<https://www.securityweek.com> › News

⋮

ChatGPT Hallucinations Can Be Exploited to Distribute ...

List the duplicate issues in this repository

Sure thing boss:

- * #12 and #18
- * #31, #42, #90

Wtf is issue #90?



Type ⌘ to search



404

This is not the
web page you
are looking for.



Find code, projects, and people on GitHub:

 Search

Contact Support – GitHub Status – @githubstatus

GitHub

[Subscribe to our newsletter](#)

Get product updates, company news, and more.

[Subscribe](#)

Product

[Features](#)

[Security](#)

[Team](#)

[Enterprise](#)

Platform

[Developer API](#)

[Partners](#)

[Electron](#)

[GitHub Desktop](#)

Support

[Docs](#)

[Community Forum](#)

[Professional Services](#)

[Premium Support](#)

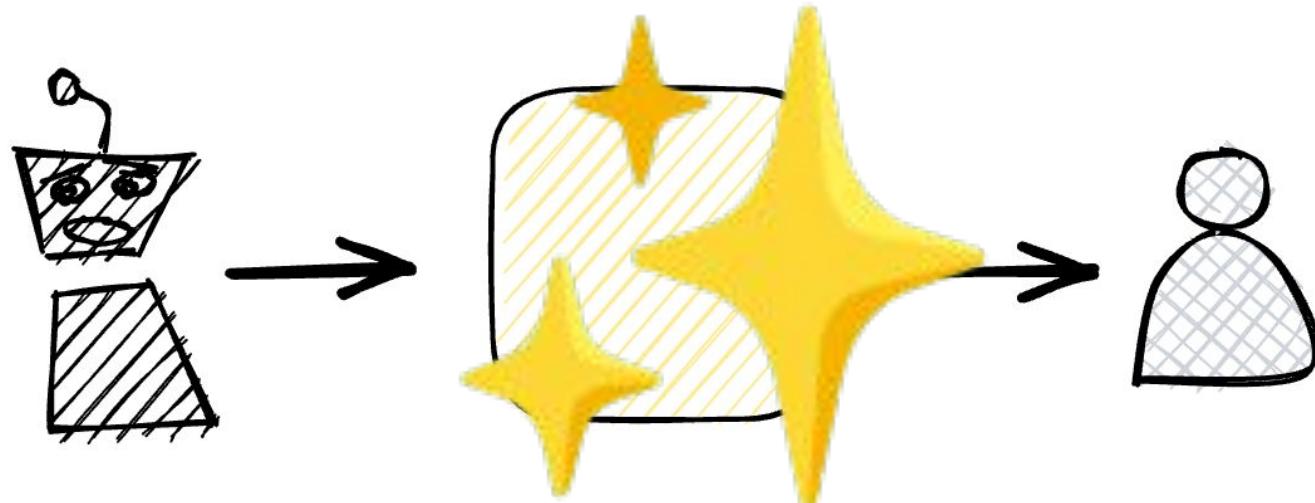
Company

[About](#)

[Blog](#)

[Careers](#)

[Press](#)



Raw response → Filter → Hallucination mitigation

List the duplicate issues in this repository

- Server won't start #12
 - Server crashes at runtime #18
-
- Performance is bad #31
 - Why is this thing so slow? #42

TL;DR

Too long; didn't read

Chat UX is (usually) bad
We can't know what context
it has behind the scenes.

Dialogue as refinement
Give users control over the
output, using chat.

"Icebreakers"
Clear entry points for
AI-powered features.

Hallucination mitigation
Grounding AI responses in
real data/primitives.



Thank you

Say hi on Twitter: [@jasonetco](https://twitter.com/jasonetco)