SMU
SINGAPORE MANAGEMENT UNIVERSITY

揚州大學
YANGZHOU UNIVERSITY

Users' Perceptions of AI Coding Assistants

Yunbo Lyu 吕允博
PhD Candidate at Singapore

July 17th, 2025, at Yang Zhou University

Integrated Development Environment (IDE)

What IDE do you currently use?

Integrated Development Environment (IDE)

IntelliJ IDEA

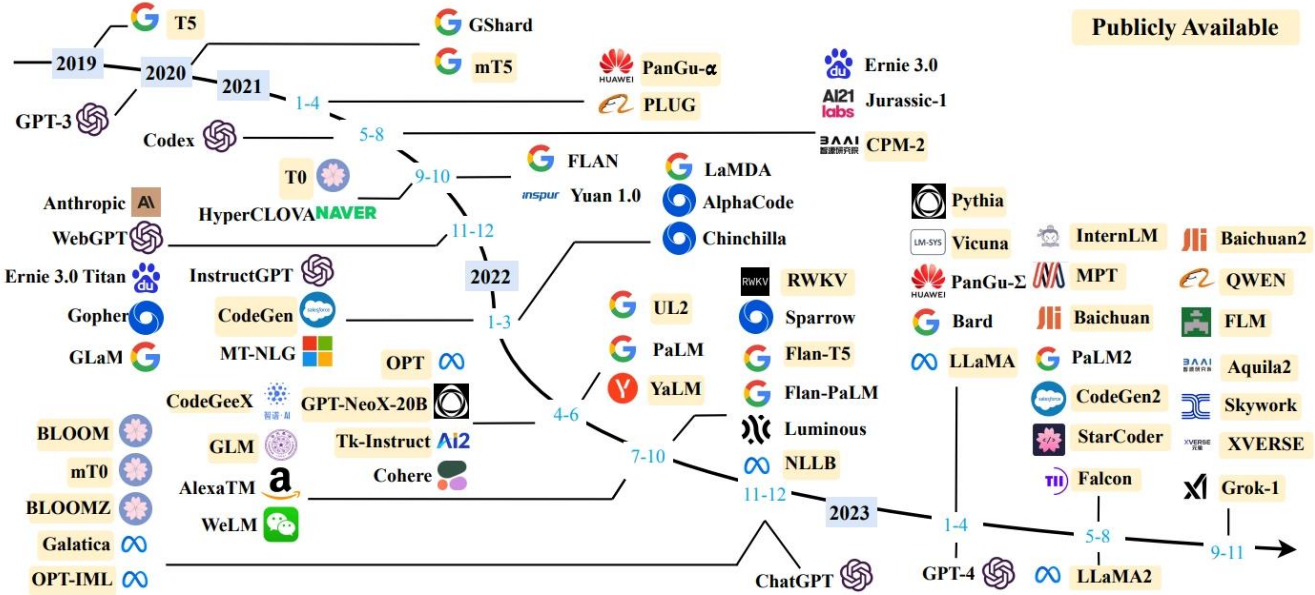
JETBRAINS IDE

Mature with the AI Coding Assistants

GitHub Copilot

October 2021

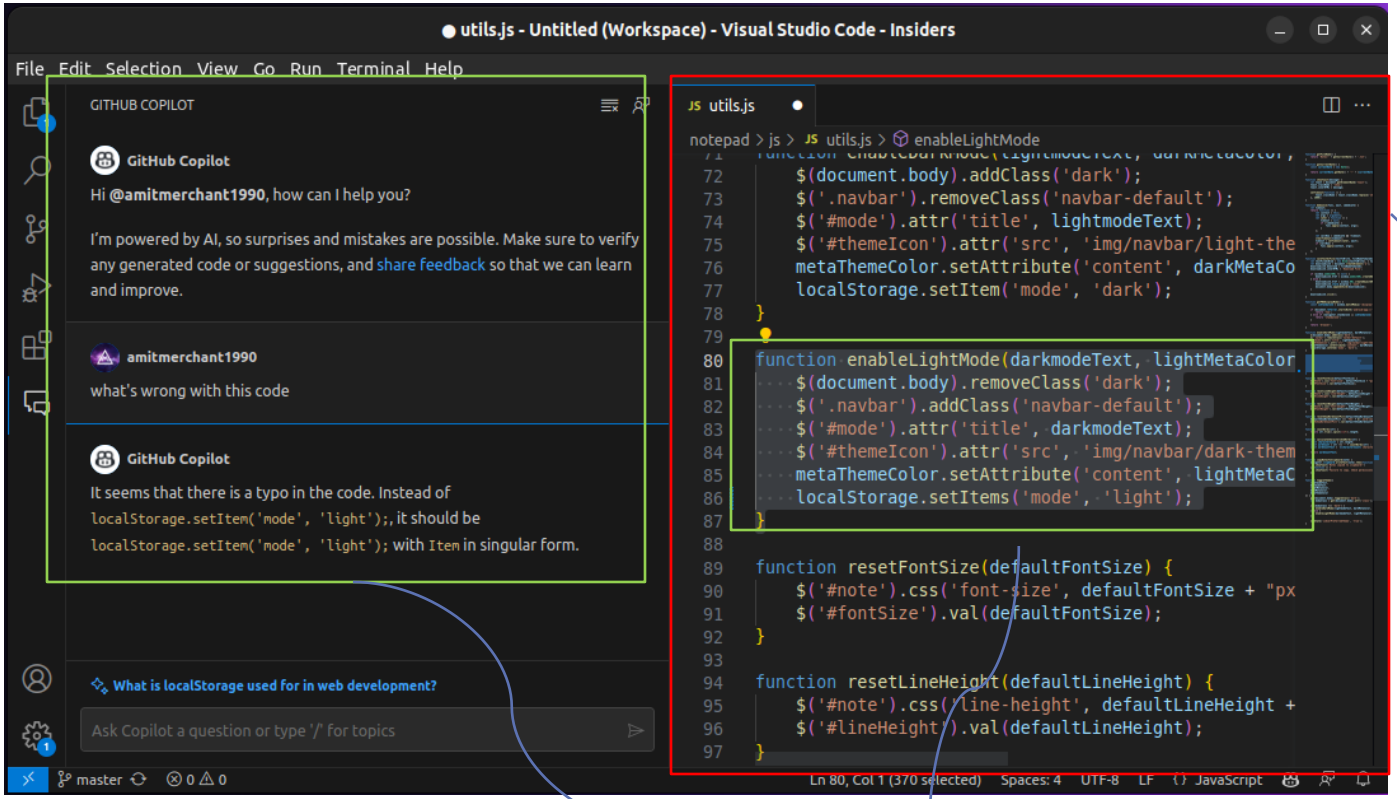
November 2022



School of Computing and Information Systems

SMU SINGAPORE MANAGEMENT UNIVERSITY

In-IDE AI Coding Assistant



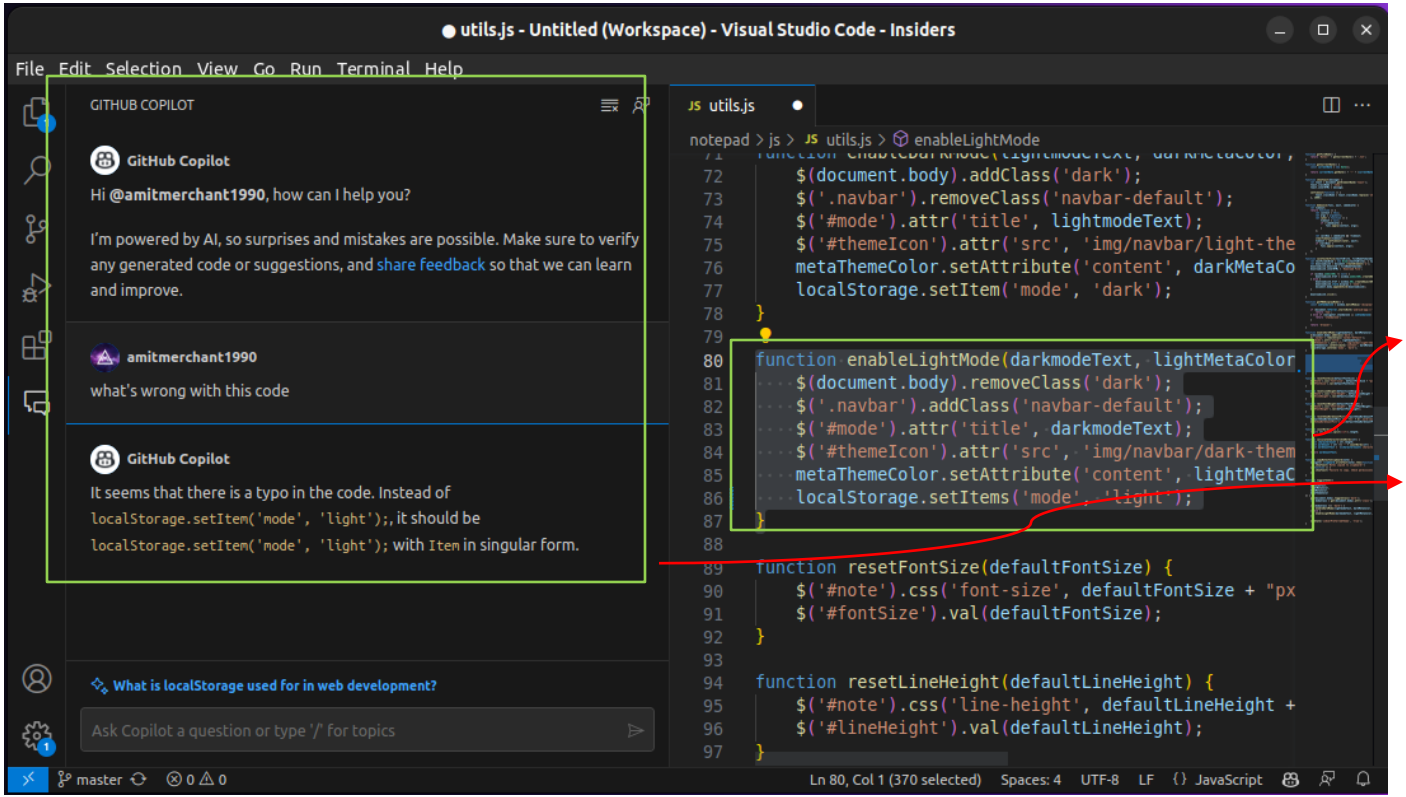
Coding Area

Interact with the AI coding assistant

School of Computing and Information Systems

SMU SINGAPORE MANAGEMENT UNIVERSITY

In-IDE AI Coding Assistant



Code Completion

Chat Interface

Agent Mode

In-IDE AI Coding Assistant

GitHub Copilot

October 2021

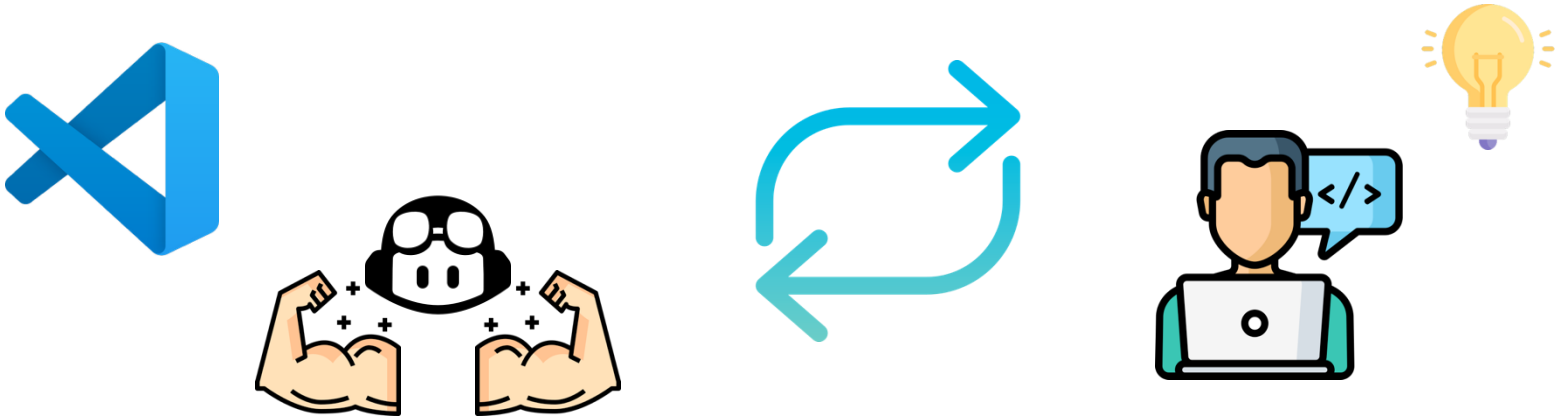
November 2022

Table 1. Codex, GPT-Neo, & TabNine evaluations for HumanEval. We find that GPT-J pass@1 is between Codex-85M and Codex-300M performance.

	PASS@k		
	$k = 1$	$k = 10$	$k = 100$
GPT-NEO 125M	0.75%	1.88%	2.97%
GPT-NEO 1.3B	4.79%	7.47%	16.30%
GPT-NEO 2.7B	6.41%	11.27%	21.37%
GPT-J 6B	11.62%	15.74%	27.74%
TABNINE	2.58%	4.35%	7.59%
CODEX-12M	2.00%	3.62%	8.58%
CODEX-25M	3.21%	7.1%	12.89%
CODEX-42M	5.06%	8.8%	15.55%
CODEX-85M	8.22%	12.81%	22.4%
CODEX-300M	13.17%	20.37%	36.27%
CODEX-679M	16.22%	25.7%	40.95%
CODEX-2.5B	21.36%	35.42%	59.5%
CODEX-12B	28.81%	46.81%	72.31%

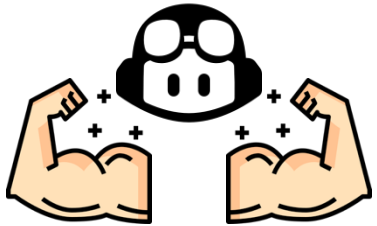
Perform good on the benchmark

Question



With the boost of AI Coding Assistants, how do users perceive them?

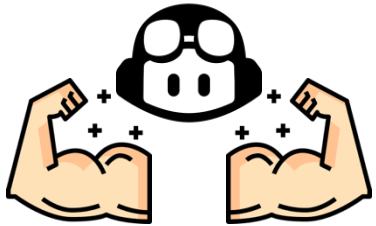
Question



With the boost of AI Coding Assistants, how do users *perceive* them?

Boost in productivity?

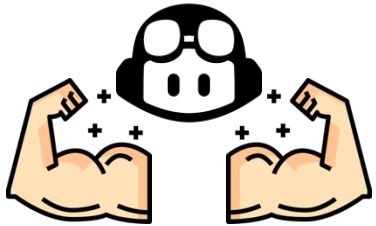
Question



With the boost of AI Coding Assistants, how do users *perceive* them?

Find a Solution?

Question



With the boost of AI Coding Assistants, how do users *perceive* them?

Distracted?

Both in Academia the Industry Curious

GitHub Copilot

October 2021

Expectation vs. Experience: Evaluating the Usability of Code Generation Tools Powered by Large Language Models

Priyan Vaithilingam
pvaithilingam@g.harvard.edu
Harvard University
USA

Tianyi Zhang
tianyi@purdue.edu
Purdue University
USA

Elena L. Glassman
glassman@seas.harvard.edu
Harvard University
USA

CHI 2022 Citations: 786

Vaithilingam, Priyan, Tianyi Zhang, and Elena L. Glassman. "Expectation vs. experience: Evaluating the usability of code generation tools powered by large language models." In *Chi conference on human factors in computing systems extended abstracts*, pp. 1-7. 2022.

1. Expectation vs. Experience

Problem

AI coding assistants' real-world usability and how they fit into a developer's workflow.

Method

- User study (N=24) comparing Copilot vs. VS Code's IntelliSense across three Python tasks (easy/medium/hard).
- Measured task success, completion time, and subjective preferences.

1. Expectation vs. Experience – Key Findings

No significant boost in completion time or success rate with Copilot vs. IntelliSense.

Strong preference for Copilot (19/24), because it:

- Provides a useful "jump-start" snippet instead of a blank editor
- Cuts down on web searches for boilerplate code

Usability hurdles hinder effectiveness:

- Difficult to **understand**, **debug,** and **edit** large AI-generated code blocks
- Cognitive overload when navigating multi-line suggestions

2. Evidence from GitHub Copilot

The Impact of AI on Developer Productivity:
Evidence from GitHub Copilot

Sida Peng,[1*] Eirini Kalliamvakou,[2] Peter Cihon,[2] Mert Demirer[3]

[1]Microsoft Research, 14820 NE 36th St, Redmond, USA
[2]GitHub Inc., 88 Colin P Kelly Jr St, San Francisco, USA
[3]MIT Sloan School of Management, 100 Main Street Cambridge, USA

GitHub 2023, Citations: 582

Experimental Design

- Controlled trial (N=95) on Upwork
- Task: Implement an HTTP server in JavaScript as fast as possible
- Treatment group used Copilot, control group used only web search/Stack Overflow

Research Question: How much can GitHub Copilot boost professional developers' productivity?

Findings: **55.8% faster task completion with Copilot**

Peng, Sida, Eirini Kalliamvakou, Peter Cihon, and Mert Demirer. "The impact of ai on developer productivity: Evidence from github copilot." *arXiv preprint arXiv:2302.06590* (2023).

3. Grounded Copilot

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

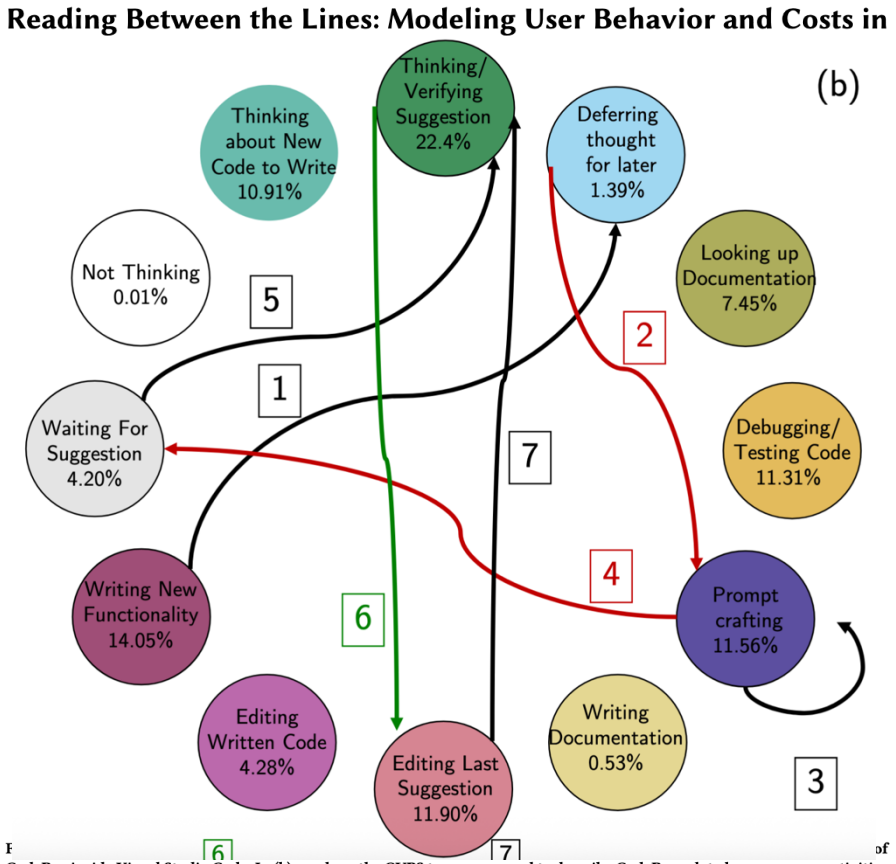
OOPSLA 2023 Citations: 458

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.

Barke, Shraddha, Michael B. James, and Nadia Polikarpova. "Grounded copilot: How programmers interact with code-generating models." *Proceedings of the ACM on Programming Languages* 7, no. OOPSLA1 (2023): 85-111.

4. Reading Between the Lines



Goals

Introduce and validate CUPS (CodeRec User Programming States), a fine-grained taxonomy and behavioral model of how developers interact with AI code recommendation tools (e.g., Copilot).

Method

Grounded-theory labeling: 21 programmers retroactively annotate 3,137 "telemetry segments" of their real Copilot sessions

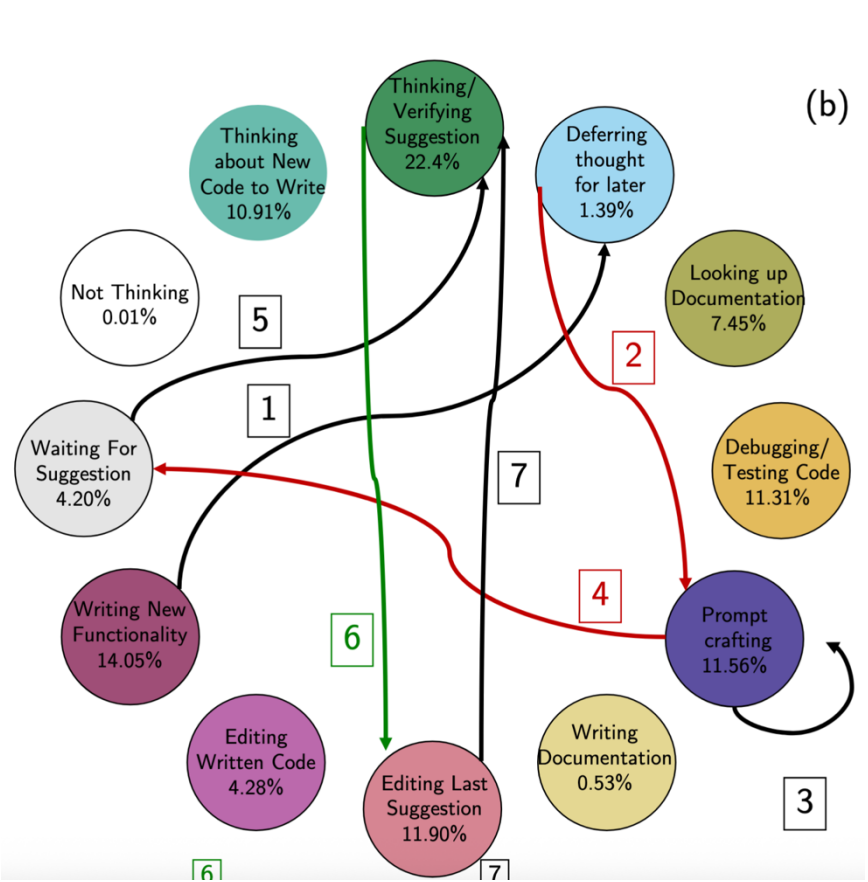
CUPS taxonomy defines 12 states (e.g., *Prompt Crafting, Verifying Suggestion, Writing New Functionality*)

School of Computing and Information Systems

CHI 2024 Citations: 147

SMU SINGAPORE MANAGEMENT UNIVERSITY

4. Reading Between the Lines – Key Findings



Verification dominates:
"Thinking/Verifying Suggestion" alone consumes **22.4%** of session time

Half the session (51.5%) was spent in Copilot-specific states (prompting, deferring, waiting, editing suggestions)

Deferred verification is common: many acceptances are immediately followed by post-accept reviews, inflating true acceptance costs

School of Computing and Information Systems

Mozannar, Hussein, Gagan Bansal, Adam Fourney, and Eric Horvitz. "Reading between the lines: Modeling user behavior and costs in AI-assisted programming." In *Proceedings of the 2024 CHI Conference on Human Factors in Computing Systems*, pp. 1-16. 2024.

SMU SINGAPORE MANAGEMENT UNIVERSITY

5. A Large-Scale Survey

A Large-Scale Survey on the Usability of AI Programming Assistants: Successes and Challenges

Jenny T. Liang
Carnegie Mellon University
Pittsburgh, PA, USA
jtliang@cs.cmu.edu

Chenyang Yang
Carnegie Mellon University
Pittsburgh, PA, USA
cyang3@cs.cmu.edu

Brad A. Myers
Carnegie Mellon University
Pittsburgh, PA, USA
bam@cs.cmu.edu

ICSE 24 Citations: 197

Core Goal

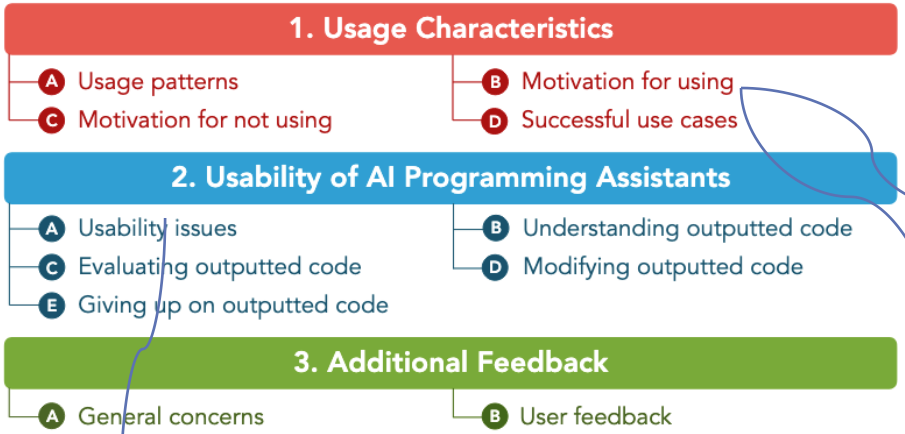
Understand, at scale, **why** developers choose (or avoid) AI programming assistants and **what** usability challenges they face

Method

Surveyed **410** real-world developers across Copilot, Tabnine, ChatGPT, CodeWhisperer, etc., combining quantitative rankings and open-ended feedback

Liang, Jenny T., Chenyang Yang, and Brad A. Myers. "A large-scale survey on the usability of ai programming assistants: Successes and challenges." In *Proceedings of the 46th IEEE/ACM international conference on software engineering*, pp. 1-13. 2024.

5. A Large-Scale Survey – Key Findings



Motivation for using

- **Autocomplete & keystroke reduction** (86%)
- **Speed up tasks** (76%)
- **Recall syntax without web search** (68%)

Top usability issues

- **What input led to this suggestion?** (30% often)
- **Giving up and rewriting** tool-generated code (28%)
- Code generation tool's suggestions **are too distracting** (23% often)

Motivation for not using:

- Generated code **fails to meet requirements** (54%)
- Hard to **control** what the tool outputs (48%)

School of Computing and Information Systems

Liang, Jenny T., Chenyang Yang, and Brad A. Myers. "A large-scale survey on the usability of ai programming assistants: Successes and challenges." In *Proceedings of the 46th IEEE/ACM international conference on software engineering*, pp. 1-13. 2024.

SMU Singapore Management University

6. Using AI-based coding assistants in practice

Using AI-based coding assistants in practice: State of affairs, perceptions, and ways forward

Agnia Sergeyuk [a,1], Yaroslav Golubev [a,*,1], Timofey Bryksin [b], Iftekhar Ahmed [c]

[a] JetBrains Research, Belgrade, Serbia
[b] JetBrains Research, Limassol, Cyprus
[c] University of California, Irvine, CA, United States

Core Goal

Conduct a **large-scale survey** (N = 481) to map exactly **where**, **how**, and **why** developers use (or avoid) AI coding assistants across the full software development lifecycle

Main Findings:

Implementing new features, is the most **enjoyable** and the least likely to be delegated to an assistant, while **writing tests and writing natural-language artifacts** are the most **unpleasant** and the most likely to be delegated.

Main Findings: Activities and their stages:

(1) Implementing new features; (2) Writing tests; (3) Bug triaging; (4) Refactoring; (5) Writing natural-language artifacts

7. Problems, Causes and Solutions

Exploring the problems, their causes and solutions of AI pair programming: A study on GitHub and Stack Overflow☆

Xiyu Zhou [a], Peng Liang [a,*], Beiqi Zhang [a], Zengyang Li [b], Aakash Ahmad [c], Mojtaba Shahin [d], Muhammad Waseem [e]

[a] School of Computer Science, Wuhan University, Wuhan, China
[b] School of Computer Science & Hubei Provincial Key Laboratory of Artificial Intelligence and Smart Learning, Central China Normal University, Wuhan, China
[c] School of Computing and Communications, Lancaster University Leipzig, Leipzig, Germany
[d] School of Computing Technologies, RMIT University, Melbourne, Australia
[e] Faculty of Information Technology, University of Jyväskylä, Jyväskylä, Finland

Core goal: Systematically characterize the real-world problems, their root causes, and practical solutions encountered by developers using GitHub Copilot as an "AI pair programmer."

Data sources: 473 GitHub Issues, 706 GitHub Discussions, and 142 Stack Overflow posts, qualitatively analyzed via grounded coding into taxonomies of problems, causes, and fixes.

7. Problems, Causes and Solutions

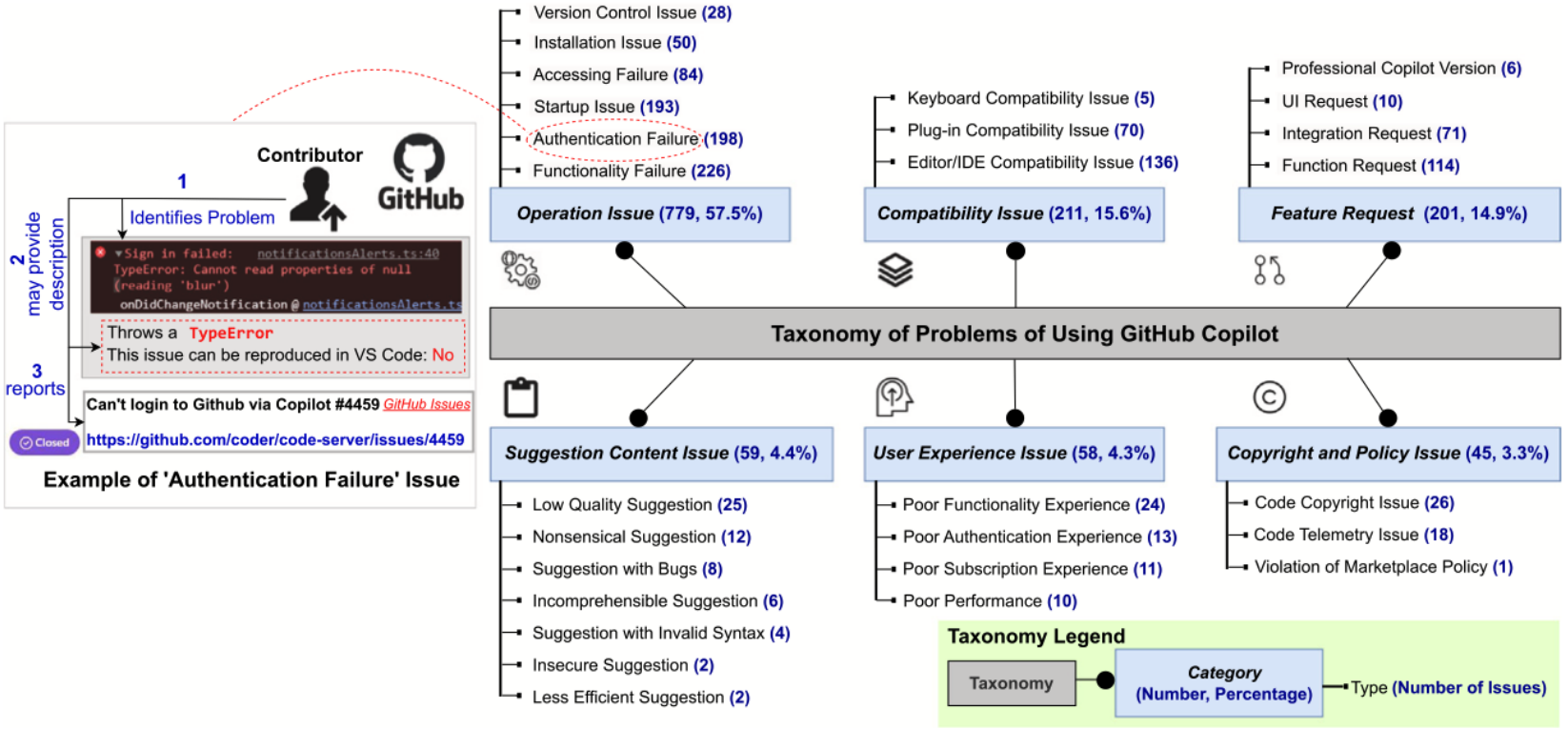
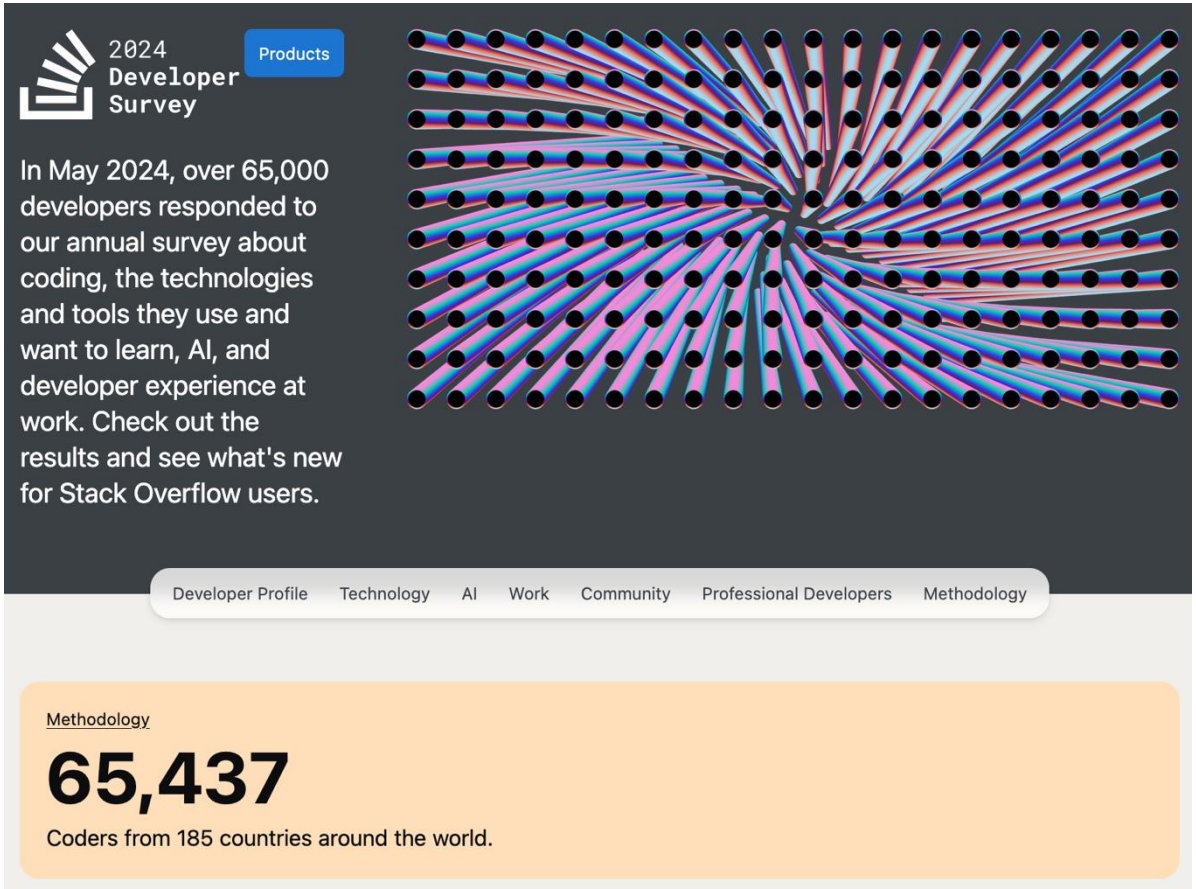


Fig. 3. A taxonomy of problems when using GitHub Copilot.

"My productivity is boosted, but ..." Demystifying Users' Perception on AI Coding Assistants

Under Submission

Integrated Development Environment (IDE)



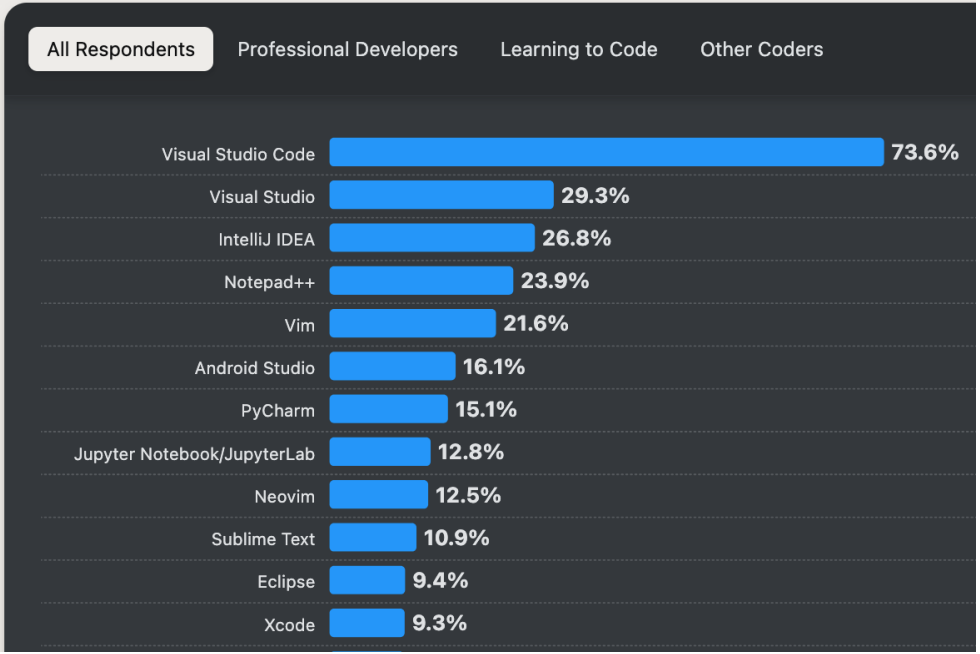
https://survey.stackoverflow.co/2024/

Integrated Development Environment (IDE)

Integrated development environment

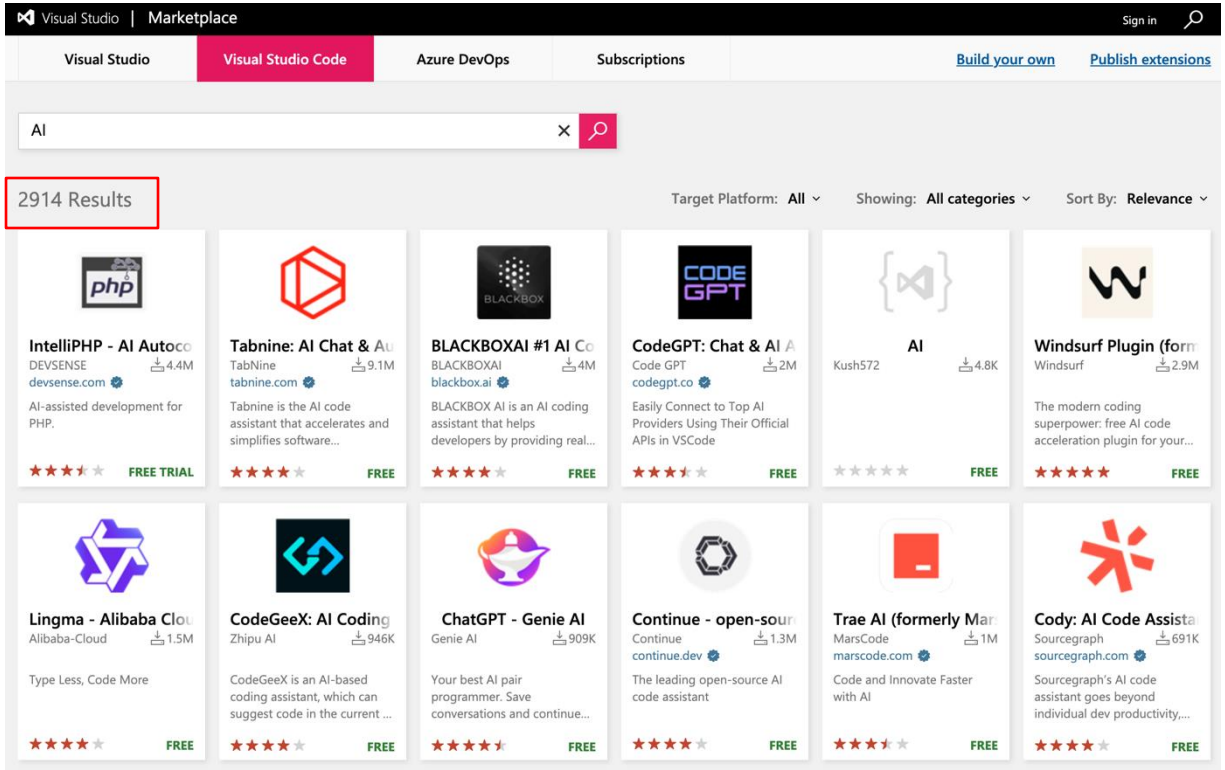
Visual Studio Code is used by more than twice as many developers than its nearest (and related) alternative, Visual Studio.

Which **development environments** did you use regularly over the past year, and which do you want to work with over the next year? Please check all that apply.



73.6% of developers use VS Code as their primary IDE.

VS Code Marketplace

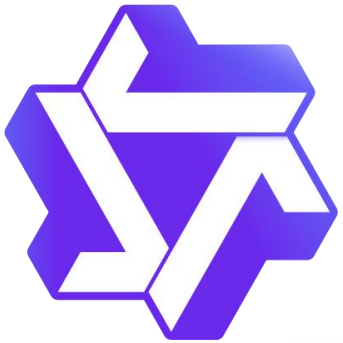


Thousands of AI Coding assistants in the VS Code marketplace.

AI Coding Assistant

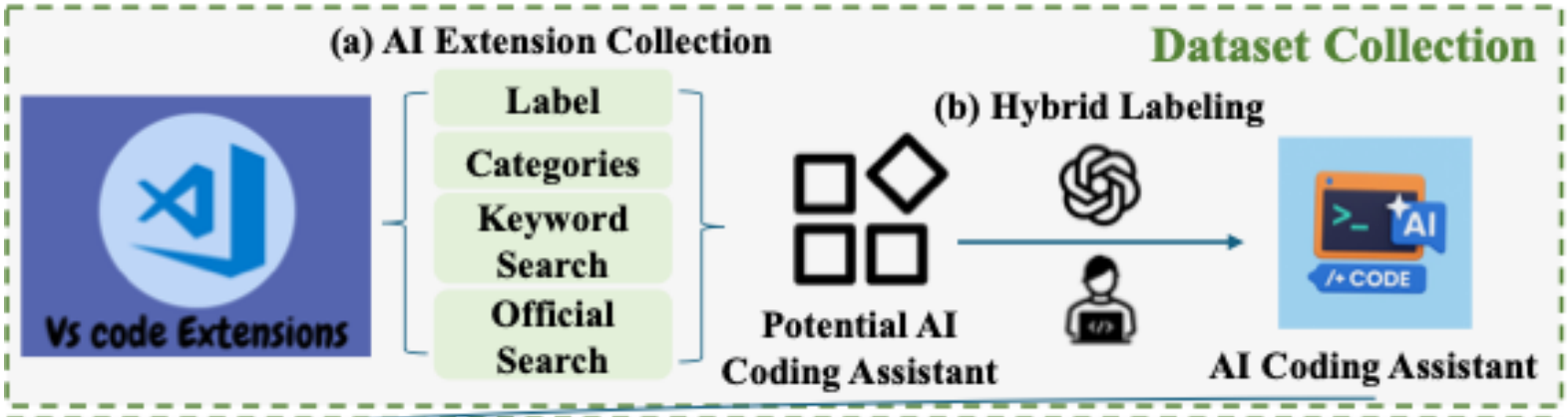
GitHub Copilot

tabnine



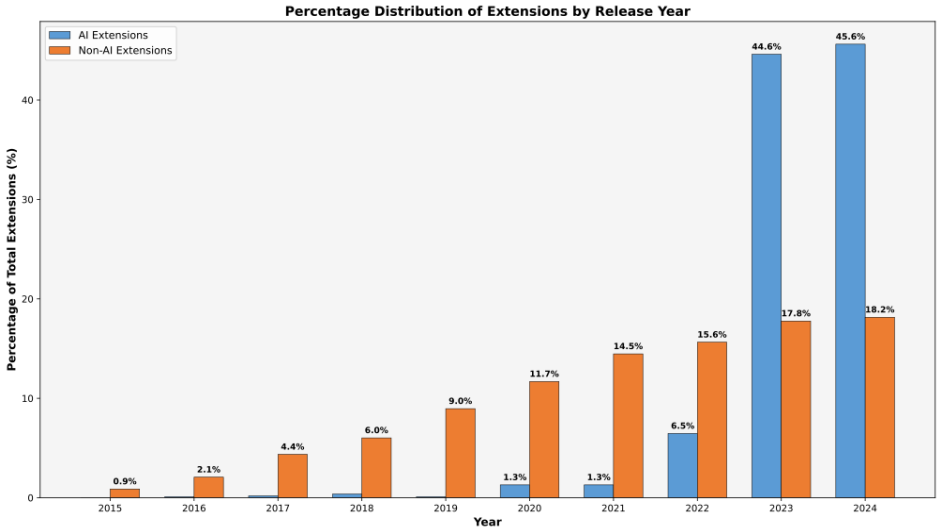
Collecting the AI Coding Assistants

96.37% precision
96.88% recall



66,053 **1,962** **1,085**

VS Code AI Coding Assistants



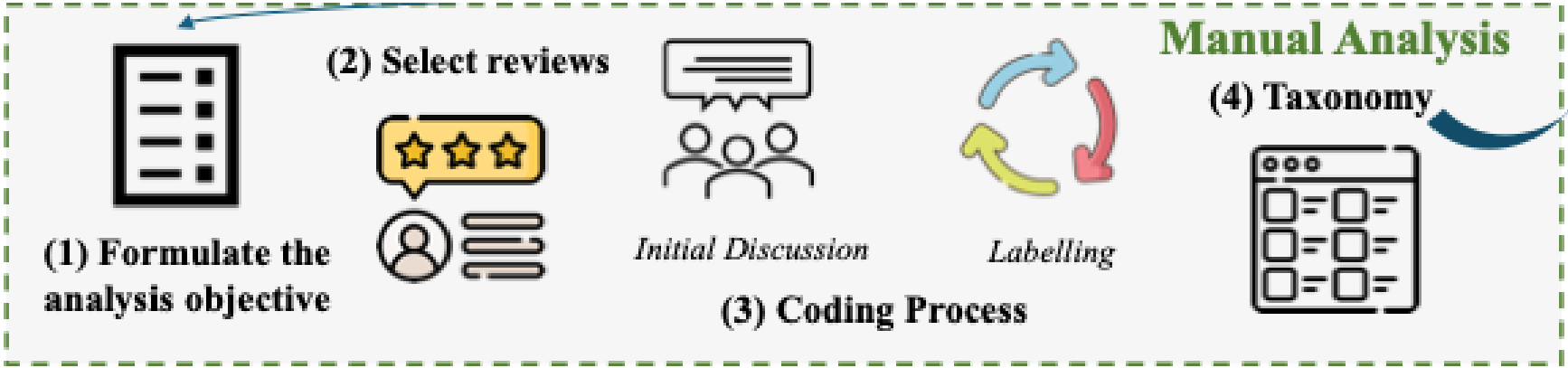
AI extensions have seen **rapid growth** in recent years

1.64% of all extensions on the VS Code Marketplace

AI extensions **receive significantly more feedback**, with an average of 7.48 ratings per extension compared to 1.76 for non-AI extensions

Both AI and non-AI extensions show similar **inequality in installs and ratings**: The top 10 most-installed AI extensions account for 86% of total installs, and the top 30 most-rated receive 75% of all user ratings

Labeling Taxonomy

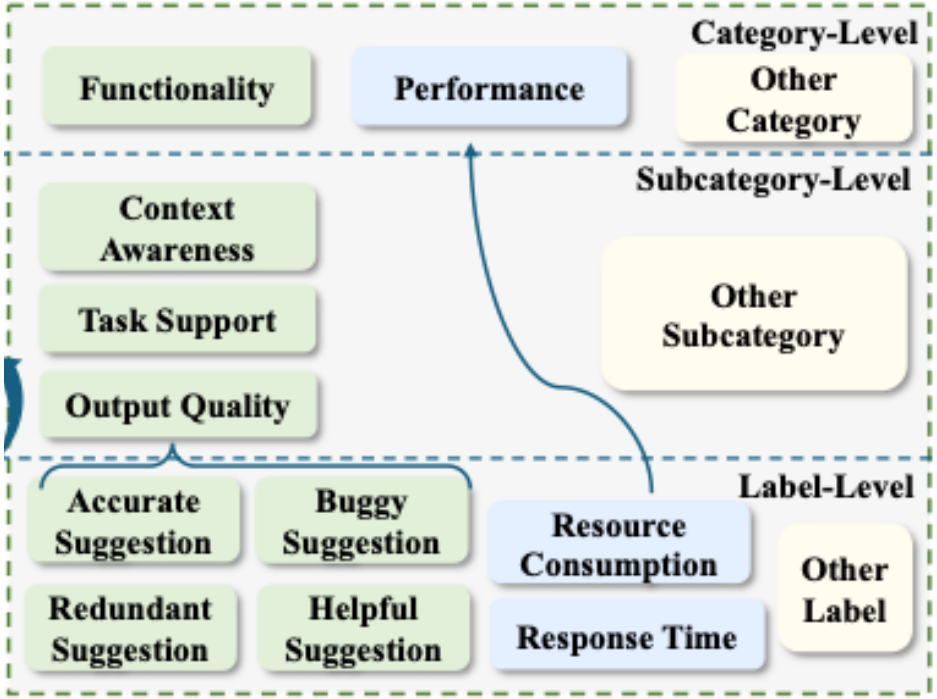


Sampled **361** user reviews from **32** popular assistants.

Conduct a **Hybrid card sorting**:

- Started with **five predefined top-level categories**
- Then use **bottom-up consolidation**
- **Iterative Coding**

Labeling Taxonomy



Developed a 3-level taxonomy (8 categories, 16 subcategories, 62 labels).

Taxonomy

ID	Category-Subcategory	Description	No.	Rate	Sentiment
1	**Functionality**	Core code-generation and assistance features.	238	32.2%	64% 31%
	• Suggestion Content	Opinions on code suggestion content.	101	13.7%	59% 37%
	• PL, Library, Task Support.	Support for languages, libraries, and SE tasks.	73	9.9%	78% 17%
	• Understanding ability	Ability to understand code and user intent.	45	6.1%	62% 29%
	• Context Awareness	Ability to leverage code/project context.	21	2.8%	38% 57%
	• DEI	Integration with tools (terminal, debugger).	7	1.0%	100% 0%
2	**General Experience**	Overall experience and emotional response.	136	18.4%	90% 10%
	• Productivity	Reported acceleration or slow down of coding speed, flow.	83	11.2%	90% 10%
	• General Discussion	Open-ended reflections or pure praise and claims.	28	3.8%	89% 11%
	• Helpfulness	Usefulness of suggestions solving problems.	23	3.4%	92% 8%
3	**Usability**	Interface design and ease of use.	104	14.1%	53% 36%
	• UI & Interactivity	Interface layout, chat panel, pop-up quality, cursor control.	60	8.1%	55% 30%
	• Controllability	Settings, model choice, interruption control.	21	2.8%	76% 19%
	• Learnability	On-boarding difficulty and docs clarity.	16	2.2%	38% 50%
	• Predictability	Consistency of responses, avoids corruption.	7	1.0%	0% 100%
4	**Dependability**	Trustworthiness: reliability, security, ethics, uptime.	83	11.2%	19% 77%
	• Reliability	Stability, crashes, install failures, fallbacks.	37	5.0%	13% 84%
	• Legal & Ethical Concerns	License compliance, AI-ethics concerns.	23	3.1%	22% 74%
	• Security & Privacy	Risk of code leaks, privacy-safeguard adequacy.	12	1.6%	17% 83%
	• Availability	Offline capability and regional limitations.	10	1.4%	40% 50%
5	**Pricing**	Monetary cost, free tiers, perceived value.	55	7.4%	69% 29%
	• Free to use	Positive reactions to generous free tiers.	32	4.3%	97% 0%
	• Value Perception	Overpricing claims versus fair-price praise.	23	3.1%	68% 32%
6	**Supportability**	Post-deployment support, compatibility, rollout.	48	6.5%	58% 33%
	• Compatibility	OS / IDE / web compatibility, conflicts.	18	2.4%	47% 35%
	• Serviceability	Vendor or community support responsiveness.	15	2.0%	93% 7%
	• Feature Availability	Speed and openness of new-feature access.	11	1.5%	46% 46%
	• Maintainability	Stability across updates, maintenance pace.	4	0.5%	20% 80%
7	**Comparison**	Comparisons with other AI tools.	44	6.0%	71% 27%
	• With Competition	General comparisons to rival products.	21	2.9%	81% 19%
	• With Github Copilot	Contrasts in accuracy, speed, price vs Copilot.	19	2.6%	74% 21%
	• With GPT	Evaluations versus raw ChatGPT / GPT-4.	4	0.5%	0% 100%
8	**Performance**	Latency, throughput, resource footprint issues.	31	4.2%	58% 42%
	• Response Time	Perceived waiting time between request and output.	17	2.3%	82% 18%
	• Resource Consumption	CPU, memory, battery drain complaints.	9	1.2%	22% 78%
	• Rate Limiting	Complaints about request caps	5	0.7%	40% 60%

Like Request Dislike

What Do Users Like and Dislike?

TABLE II: What Do Users Like and Dislike?

Top-15 Users' Like			Top-15 Users' Dislike		
no.	Label	N.	no.	Label	N.
L1	Accuracy suggestion	39	D1	helpfulness suggestion	17
L2	Task support	24	D2	AI ethics	15
L3	PL Support	20	D3	Bug of the extension	12
L4	Chat interface	16	D4	complet & redundant	8
L5	helpfulness suggestion	14	D5	Resource Consumption	7
L6	Response Time	14	D6	Project Context Support	6
L7	Serviceability	14	D7	Suggestion UI	6
L8	Customization	12	D8	Chat interface	6
L9	Code understanding	11	D9	PL Support	6
L10	IDE Compatibility	8	D10	Context-memory capacity	6
L11	General design	7	D11	On-boarding Difficulty	6
L12	Project Context Support	7	D12	Mess up the code	5
L13	Framework support	6	D13	Fallback to weak model	5
L14	Suggestion UI	6	D14	Frustration waiting list	5
L15	On-boarding Difficulty	5	D15	Login Issue	5

Finding 1

1. Productivity Boost is Real—but Not Universal

- Most users report productivity gains, especially novices.
- Experienced developers are more critical.

"not having to type every single repetitive function out or imports" (R94, 5☆).

"I am a beginner programmer, and it is helping me a lot to build a project" (R319, 5☆).

"For anyone who really knows how to code, save yourself a lot of frustration" (R14, 1☆).

Finding 2

2. Suggestion Quality is the Top Concern

- Accurate suggestions are highly valued.
- Users dislike redundancy, incompleteness, and buggy outputs.

"80% less keyboard touching. Autocomplete is pure magic. Feels like it's connected directly to your mind" (R164, 5☆).

"Constantly barfs words on the screen, 90+% is repetitive." (R14, 1☆)

"it only predicts one character for me" (R34, 1☆).

Finding 3

3. Context Awareness is a Major Weakness

- Assistants can interpret code but struggle to fetch or retain context, especially at the Repository level.

"[assistant] still doesn't see the class definitions in files that aren't open" (R1, 1☆).

"[assistant] forgets context on next question and answers irrelevantly even for simple questions" (R22, 1☆).

Finding 4

4. Usability Matters

- Poor onboarding and intrusive interface elements can deter users.

"Setup process is bloated. I'll wait until they make the process more streamlined." (R265, 1☆).

"While [assistant] aims to simplify coding, some users might find it challenging to adapt to the AI's suggestions and functionality, especially if they're used to traditional coding practices." (R240, 4☆).

"Annoyed suggestions show up at the top", "Focus doesn't work, making chat useless...frustrated, don't use this extension." (R312, 1☆).

"Messed so much with my code" (R7, 3☆).

Finding 5

5. Resource Consumption is a Pain Point

- Users appreciate fast response time but complain about high CPU/memory usage.

Uses too many resources—over 50% CPU and more than 1 GB memory" (R125, 1☆).

"The extension's performance can sometimes slow down the editor, especially when working on larger files or multi-projects" (R306, 5☆).

Finding 6

6. Pricing and Ethics Influence Adoption

- Users prefer free tools and criticize the monetization of open-source trained models.

"It's a wonderful free alternative of paid AI code assistants"

"Was cool to try out but too expensive now. You are using our code to make money. So, pass for now...but I think you should have a free version (since it's using open source)" (R42, 1☆).

Recommend Reading

Systematically learn software engineering for software systems with ML components:

1. CMU 17-445/645: Machine Learning in Production (Class)

Empirical Study(Christian Kästner):

2. Collaboration Challenges in Building ML-Enabled Systems: Communication, Documentation, Engineering, and Process
3. A large-scale survey on the usability of AI programming assistants: Successes and challenges

How to do Taxonomy:

4. Taxonomy of real faults in deep learning systems

SMU SINGAPORE MANAGEMENT UNIVERSITY

揚州大學 YANGZHOU UNIVERSITY

Hope it sparks!
Questions are welcome.

Contact: 吕允博 (Yunbo Lyu)
Email:
yunbolyu@smu.edu.sg
Personal Website:
https://yunbolyu.github.io

Wechat