Towards Reliable LLM-based Software Development Tools

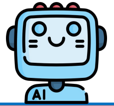
Yue Liu

Monash University, Australia

MONASH University

Large Language Models(LLMs) are Everywhere Nowadays!

- Massive Model Size
- Diverse training data
- Computational Power
- Pre-training and fine-tuning
- Transfer learning
- ....



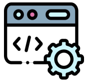
Powerful & Accurate LLMs

Writing

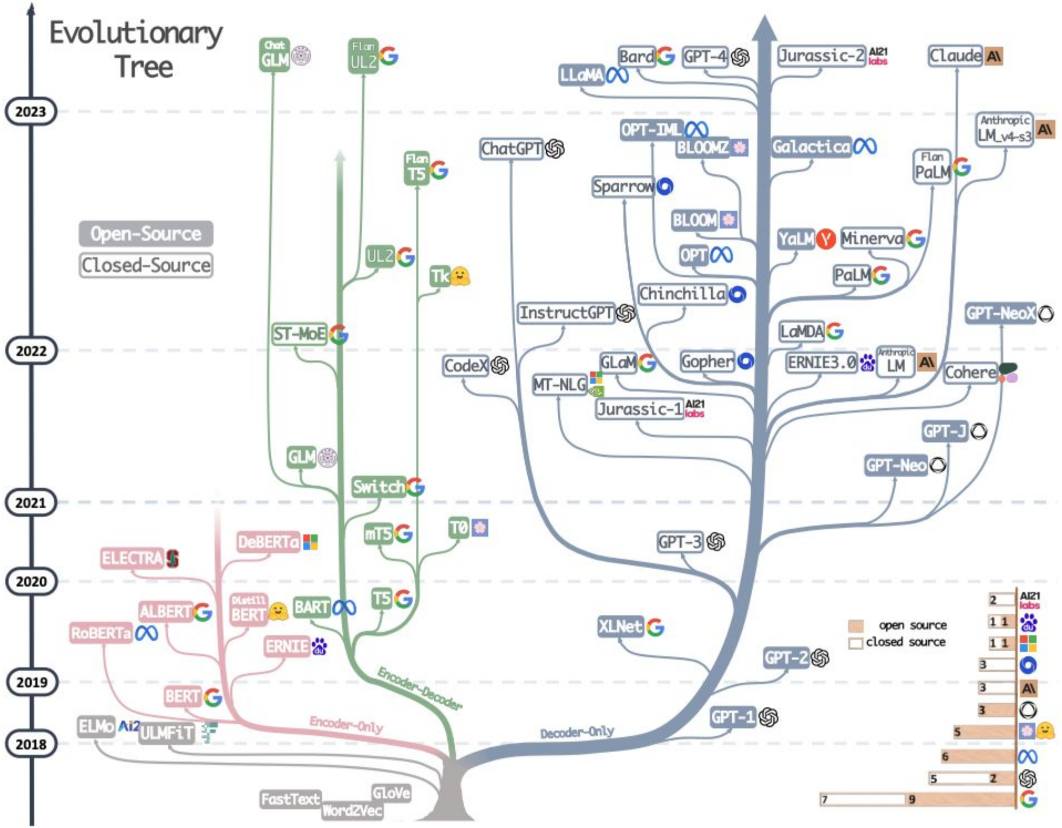
Language Translation

Answering Questions

Image Generation



Software Development



Yang J, Jin H, Tang R, et al. Harnessing the power of llms in practice: A survey on chatgpt and beyond[J]. ACM Transactions on Knowledge Discovery from Data, 2023.

Motivation Examples

In Dec 2023, Google Inc. announced **Gemini Ultra**, which set the state of the art across a wide range of benchmarks for text, image, audio, video and code (Over GPT-4 and Claude-2)

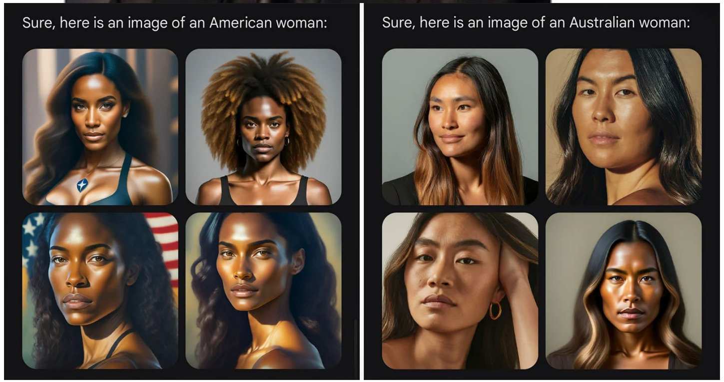
High accuracy

In Feb 2024, Google Inc. launched Gemini Ultra for users. However, for image generation feature, Gemini would sometimes ‘overcompensate’ for diversity.

Not reliable LLM-generated images

In 23 Feb 2024, Google Inc. apologized and turned off the image generation of people.

Not deployed in real-world application



LLMs Require Both High Accuracy and Robust Reliability

In Dec 2023, Google Inc. announced **Gemini Ultra**, which set the state of the art across a wide range of benchmarks for text, image, audio, video and code (Over GPT-4 and Claude-2)

High accuracy

In Feb 2024, Google Inc. launched Gemini Ultra for users. However, for image generation feature, Gemini would sometimes ‘overcompensate’ for diversity.

Not reliable LLM-generated images

In 23 Feb 2024, Google Inc. apologized and turned off the image generation of people.

Not deployed in real-world application

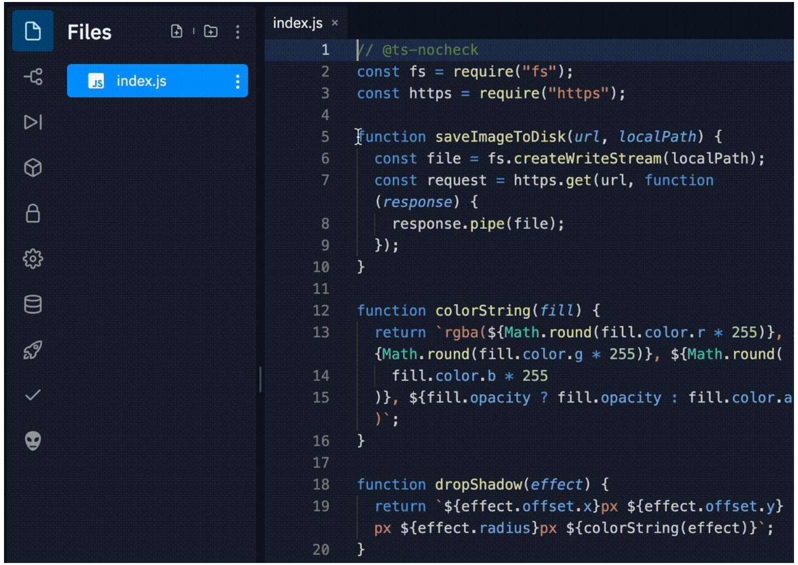
Accuracy: How much do the generated results differ from the ground truth?

Reliability (i.e., Truthfulness): The trustworthiness of results and the confidence in applying them in practical applications.

Reliability is important! Without reliable LLMs, widespread application is impossible.

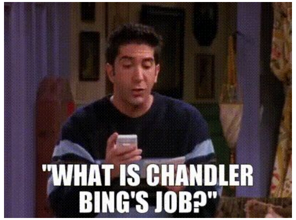
- Is the evaluation performance of LLMs trustworthy?
- Why should we trust or distrust the outputs of LLMs?
- How secure and stable is the environment to use LLMs?
- ….

LLM-based Software Development Tools



C# IntelliCode · GitHub Copilot · CODE GPT · tabnine · ChatGPT · Claude.AI · aws CodeWhisperer · Gemini

- Code Generation
- Code Repair
- Code Translation
- Code Review
- Code Completion
- Code Understanding
- Code Commit Generation
- Program Synthesis
- ……



As a data analyst, LLM-based software development tools are helping us improve productivity when developing code!!

Large Language Models for Software Engineering: A Systematic Literature Review, Xinyi Hou, Yanjie Zhao, Yue Liu, Zhou Yang, Kailong Wang, Li Li, Xiapu Luo, David Lo, John Grundy, Haoyu Wang, at arxiv 2023

Prior Research for LLM-based Software Development

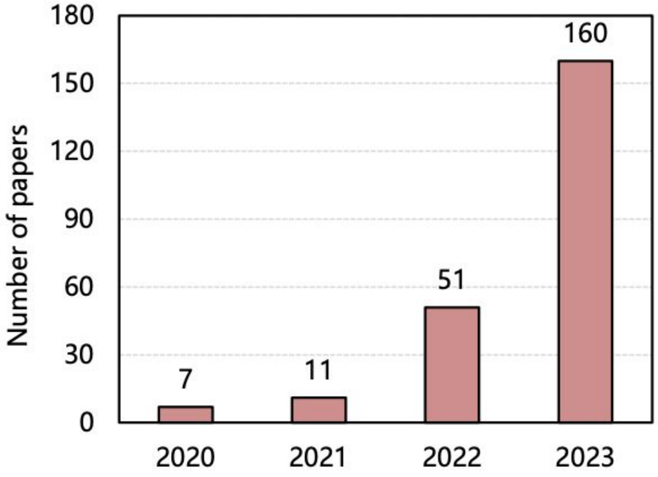
CodeT5,
UniXCoder,
CodeBERT,
CodeT5,
GPT-4
….

Software Development Tasks

Code Generation,
Bug Fixing,
Code Translation,
Code Completion,
…

Technical Improvements

- Model Architecture
- Training Techniques
- Optimization Algorithms
- Regularization

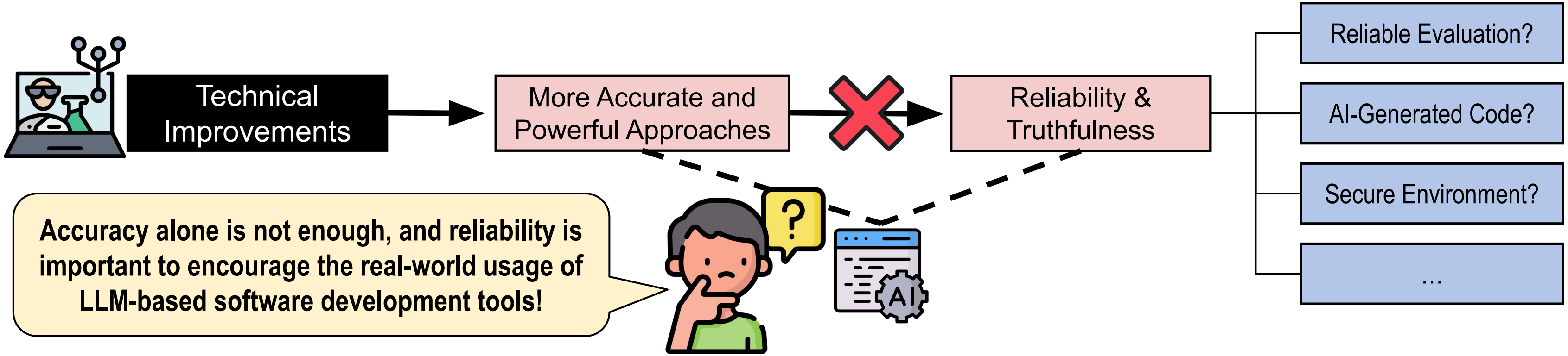


Counts of relevant research by a recent survey

While an increasing number of studies are concentrating on enhancing the accuracy of LLM-based software development through technical improvements, the aspect of reliability often remains overlooked.

Large Language Models for Software Engineering: A Systematic Literature Review, Xinyi Hou, Yanjie Zhao, Yue Liu, Zhou Yang, Kailong Wang, Li Li, Xiapu Luo, David Lo, John Grundy, Haoyu Wang, at arxiv 2023

Reliability of LLM-based Software Development Tools



Most prior research work focuses on technical improvements (e.g., model architecture improvements, training strategies, data augmentation)

- **Accuracy**: How much do the generated results differ from the ground truth?
- **Reliability** : The trustworthiness of results and the confidence in applying them in practical software development

Overarching RQ: What are the key factors/issues that could impact the reliability of LLM-based software development tools, and how do they influence their reliability?

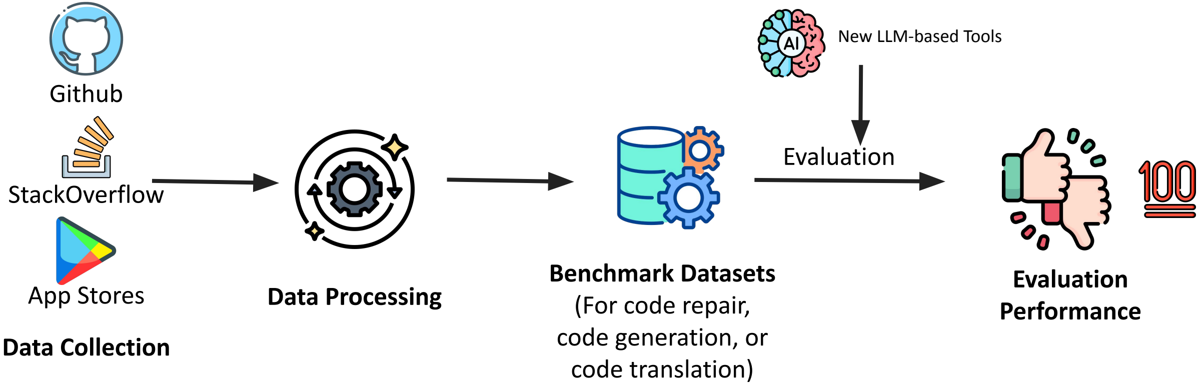
Reliability of LLM-based Software Development Tools

Reliability & Truthfulness

- 100% **Are evaluation benchmark datasets reliable?**
  - Evaluating these tools on unreliable benchmark datasets can result in misleading results
  - Investigate benchmark dataset quality and model robustness
- Ai **Is the code generated by LLMs reliable enough to be applied in the real world?**
  - LLM tools that produce low-quality code are not reliable for real-world usages
  - Investigate reliability and quality of the LLM-based generated code
- **Are LLM-based software development applications in VSCode reliable for use?**
  - Users will not trust LLM development tools that have significant security flaws
  - Investigate the security flaws in software development environment

Part I: Reliability of Evaluation Benchmark Datasets

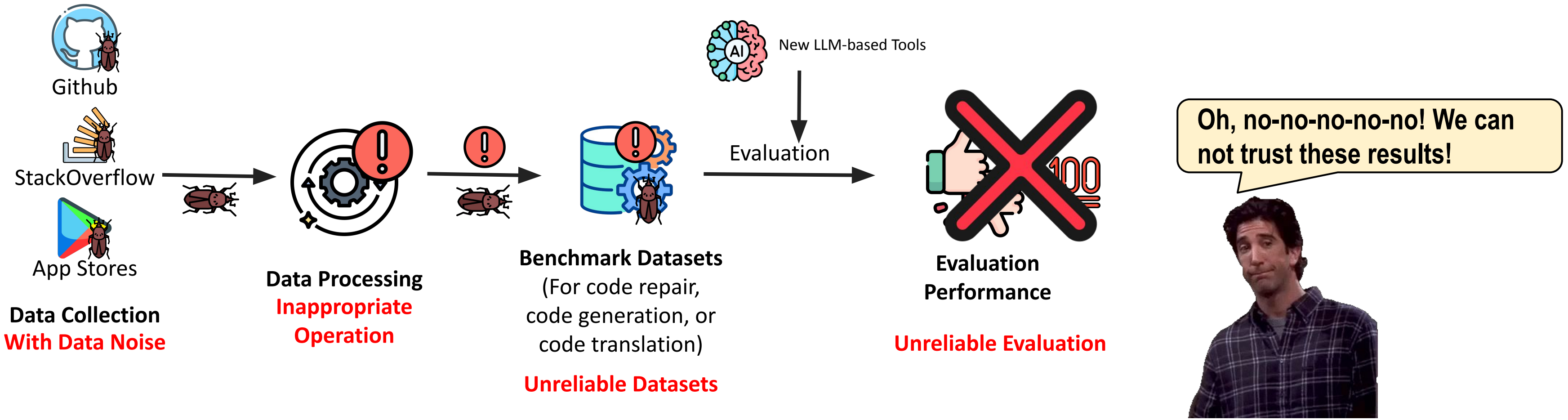
Benchmark datasets are collections of data used to evaluate and compare the performance of LLM-based software development tools. Benchmark datasets usually consist of input data, ground truth or reference labels.



Liu, Yue, et al. "On the Reliability and Explainability of Language Models for Program Generation." *ACM Transactions on Software Engineering and Methodology* (2024).

Part I: Reliability of Evaluation Benchmark Datasets

Benchmark datasets are collections of data used to evaluate and compare the performance of LLM-based software development tools. Benchmark datasets usually consist of input data, ground truth or reference labels.



RQ1: Are evaluation benchmark datasets for LLM-based software development reliable, and how do they influence the reliability?

Liu, Yue, et al. "On the Reliability and Explainability of Language Models for Program Generation." *ACM Transactions on Software Engineering and Methodology* (2024).

Part I: Reliability of Evaluation Benchmark Datasets

To answer the questions, we conducted the first comprehensive benchmark study of LLMs for program development, investigating the data duplication issues of existing evaluation benchmark datasets and and analyzing the robustness of models built on these benchmark datasets.

- **Four software development task scenarios**: code review, code repair, code translation, code generation;
- **12 benchmark datasets**: Android_S, Android_M, Google_S, Google_M, Ovirt_S, Ovirt_M, CodeReview, B2F_S, B2F_M, Java2C#, C#2Java, and CONCODE;
- **Eight large language models**: T5, CoTexT, CodeT5, CodeBERT, CodeTrans, CodeGPT, CodeReviewer, CodeT5+

Task	Subsets	Category	Language	Dataset Size
Android_S, Android_M, Google_S, Google_M, Ovirt_S, Ovirt_M	Code Review	Code-Code	Java	21,774
CodeReview	Code Review	Code+Comment-Code	Java, Python, Go, C++, C, C#, JavaScript, Php, Ruby	1.3M
B2F_S, B2F_M	Code Repair	Code-Code	Java	123,805
Java2C#, C#2Java	Code Translation	Code-Code	Java, C#	11,500
CONCODE	Code Generation	Text-Code	Java	104,000

Liu, Yue, et al. "On the Reliability and Explainability of Language Models for Program Generation." *ACM Transactions on Software Engineering and Methodology* (2024).

Part I: Experimental Finding 1

- **Data Duplications exist between training and testing sets**: 11 out 12 benchmark datasets contain over 20% of test instances that are similar to the training set, leading to exaggerated and unrealistic performance;

		Android_S	Android_M	Google_S	Google_M	Ovirt_S	Ovirt_M	CodeReview	B2F_S	B2F_M	Java2C#	C#2Java	CONCODE
Test Samples Percentage (>0.6)		53.69%	60.62%	60.88%	71.21%	71.72%	85.74%	0.05%	62.81%	21.82%	59.80%	61.20%	25.25%
CodeReviewer	Original Accuracy	14.68%	10.40%	11.81%	6.85%	25.49%	18.18%	30.43%	17.94%	8.77%	63.10%	70.40%	22.65%
	New Accuracy	13.61%	8.02%	12.61%	4.81%	25.25%	14.39%	30.44%	14.24%	7.64%	40.30%	53.87%	19.26%
	Original BLEU	0.70	0.72	0.71	0.73	0.75	0.77	0.86	0.75	0.85	0.92	0.93	0.59
	New BLEU	0.70	0.72	0.70	0.67	0.73	0.72	0.86	0.76	0.85	0.89	0.91	0.56
CodeT5+	Original Accuracy	15.20%	11.36%	14.27%	7.27%	26.06%	20.03%	30.12%	18.44%	7.84%	63.90%	70.60%	21.85%
	New Accuracy	13.09%	9.07%	13.29%	6.97%	25.13%	14.21%	30.14%	14.75%	7.11%	42.04%	53.09%	18.39%
	Original BLEU	0.70	0.72	0.72	0.72	0.75	0.77	0.85	0.75	0.85	0.93	0.93	0.59
	New BLEU	0.70	0.72	0.71	0.66	0.74	0.71	0.85	0.76	0.85	0.89	0.91	0.56

Table: Model Performance Before and After Removing High-Similarity Test Instances between Training and Testing sets

When we remove the duplicated testing instances from benchmark datasets, we observe a decrease in performance

Part I: Experimental Finding 2

- **Data Duplication across Testing Sets**: 10 out of 12 contain duplicated source sequences within their test instances, despite requiring models to generate different targets (ground truth).

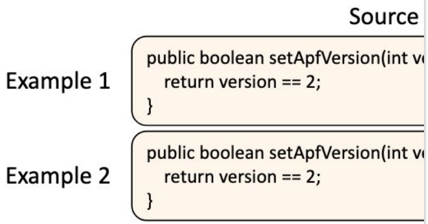
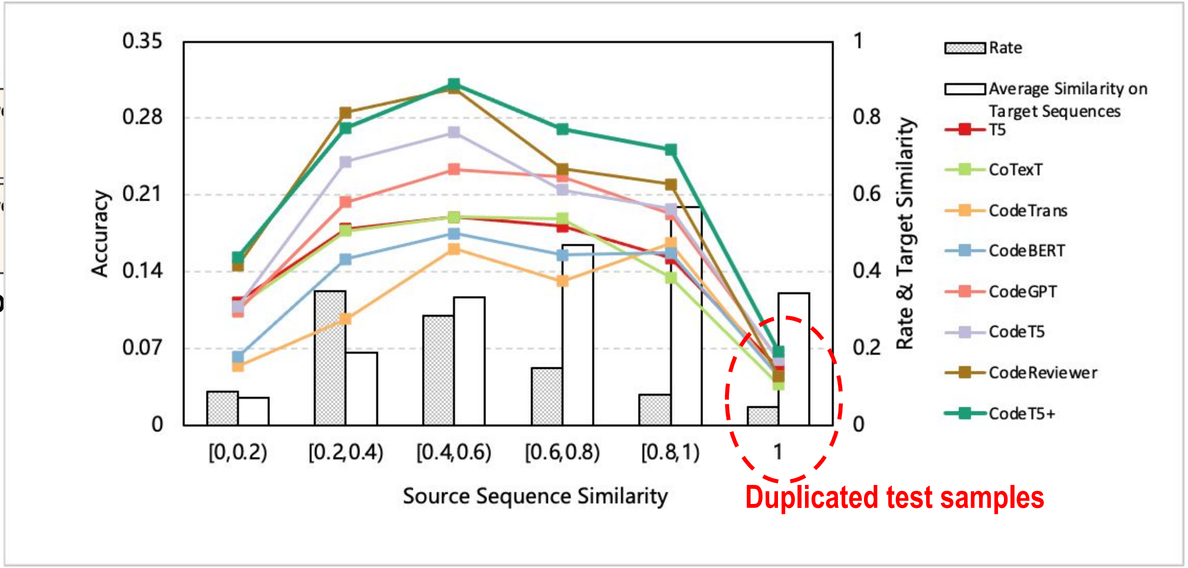


Figure: Examp



The performance on these duplicated test instances can significantly deviate from the average, potentially leading to a misrepresentation of the model's true performance.

Part I: Experimental Finding 3

Poor robustness on low-quality benchmark datasets:

- We investigated the robustness of LLMs on benchmark datasets using *SHAP*, an Explainable AI method.
- SHAP helped identify feature importance within the data. We then removed tokens with lowest importance and re-evaluated LLM accuracy.

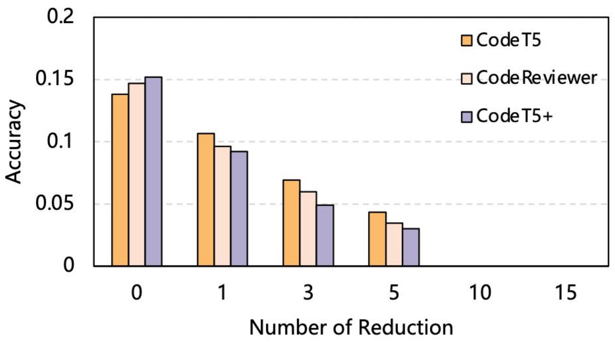


Figure: Impacts of token reduction size

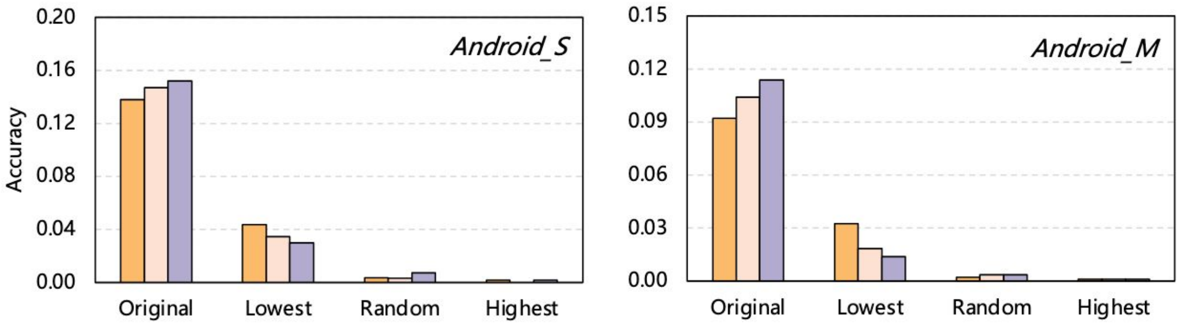


Figure: Impacts of input token reduction using different strategies (reduction size = 5)

The results revealed that the removal of even a few tokens with the lowest feature importance can lead to a significant decline in performance.

Part I: Reliability of Evaluation Benchmark Datasets

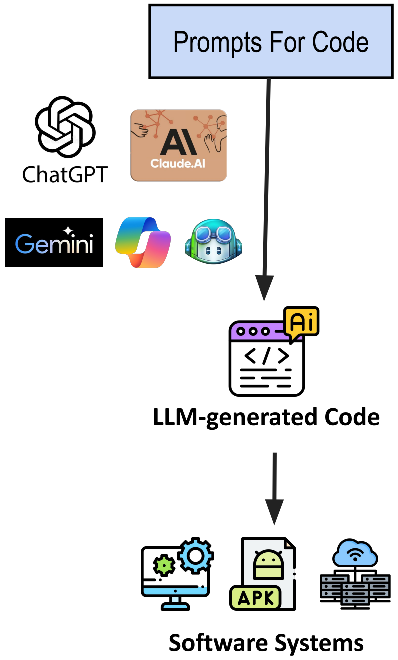
- **Data Duplications exist between training and testing sets**: 11 out 12 benchmark datasets contain over 20% of test instances that are similar to the training set, leading to exaggerated and unrealistic performance
- **Data Duplication across Testing Sets**: 10 out of 12 contain duplicated source sequences within their test instances, despite requiring models to generate different targets (ground truth).
- **Poor Robustness on Low-quality Benchmark Datasets**: the removal of even a few tokens with the lowest feature importance can lead to a significant decline in performance.

RQ1: Are evaluation benchmark datasets for LLM-based software development reliable, and how do they influence the reliability?

Answer: Data duplication and lack of diversity in benchmark datasets inflate performance metrics, leading to unreliable performance evaluations in LLM-based software development. This lack of reliability can result in poor model robustness, affecting the trustworthiness of the models.
Future work: Improve reliability and quality of benchmark datasets; Develop more robust and trustworthy evaluation methods

Liu, Yue, et al. "On the Reliability and Explainability of Language Models for Program Generation." *ACM Transactions on Software Engineering and Methodology* (2024).

Part II: Reliability of Code Generated by LLM-based Tools



So, NOWADAYS, I can just use some…LL…LLM things to write code. Could I BE any lazier?

ChatGPT 3.5

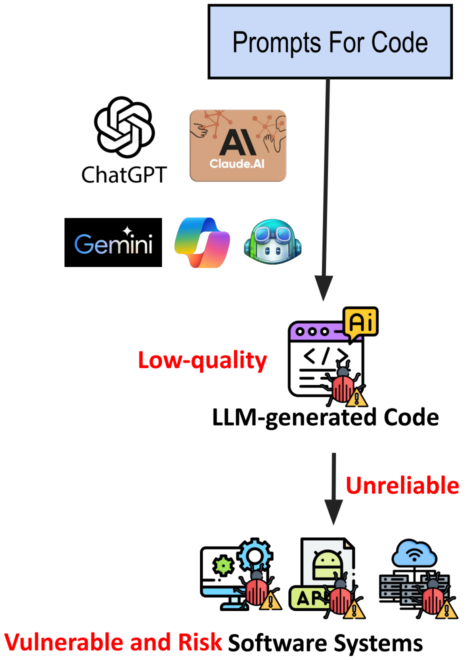
You
Write a python3 code to calculate the sum of numbers

ChatGPT
Certainly! Below is a Python 3 code that calculates the sum of numbers provided by the user:

python
```python
def calculate_sum_of_numbers():
    try:
        # Take input from the user
        numbers = input("Enter numbers separated by spaces: ")

        # Convert the input string into a list of numbers
        numbers_list = [float(num) for num in numbers.split()]

        # Calculate the sum of numbers
        total_sum = sum(numbers_list)

        # Print the result
        print("The sum of the numbers is:", total_sum)

        return total_sum

    except ValueError:
        print("Error: Please enter valid numbers separated by spaces.")
        return None
```

Liu, Yue, et al. "Refining ChatGPT-generated code: Characterizing and mitigating code quality issues." *ACM Transactions on Software Engineering and Methodology* (2024).

Part II: Reliability of Code Generated by LLM-based Tools

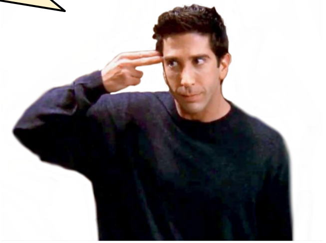


So, NOWADAYS, I can just use some…LL…LLM things to write code. Could I BE any lazier?

```java
class Solution {
    public String makeFancyString(String s) {
        int n = s.length();
        if (n < 3) {
            return s;
        }
        char[] charArr = s.toCharArray();
        for (int i = 2, j = 2; i < n; i++) {
            if (charArr[j - 2] != charArr[i]) {
                charArr[j++] = charArr[i];
            }
        }
        return new String(charArr, 0, j);
    }
}
```

Buggy Generated Java Code by ChatGPT. Variable "j" is used outside the "for" loop

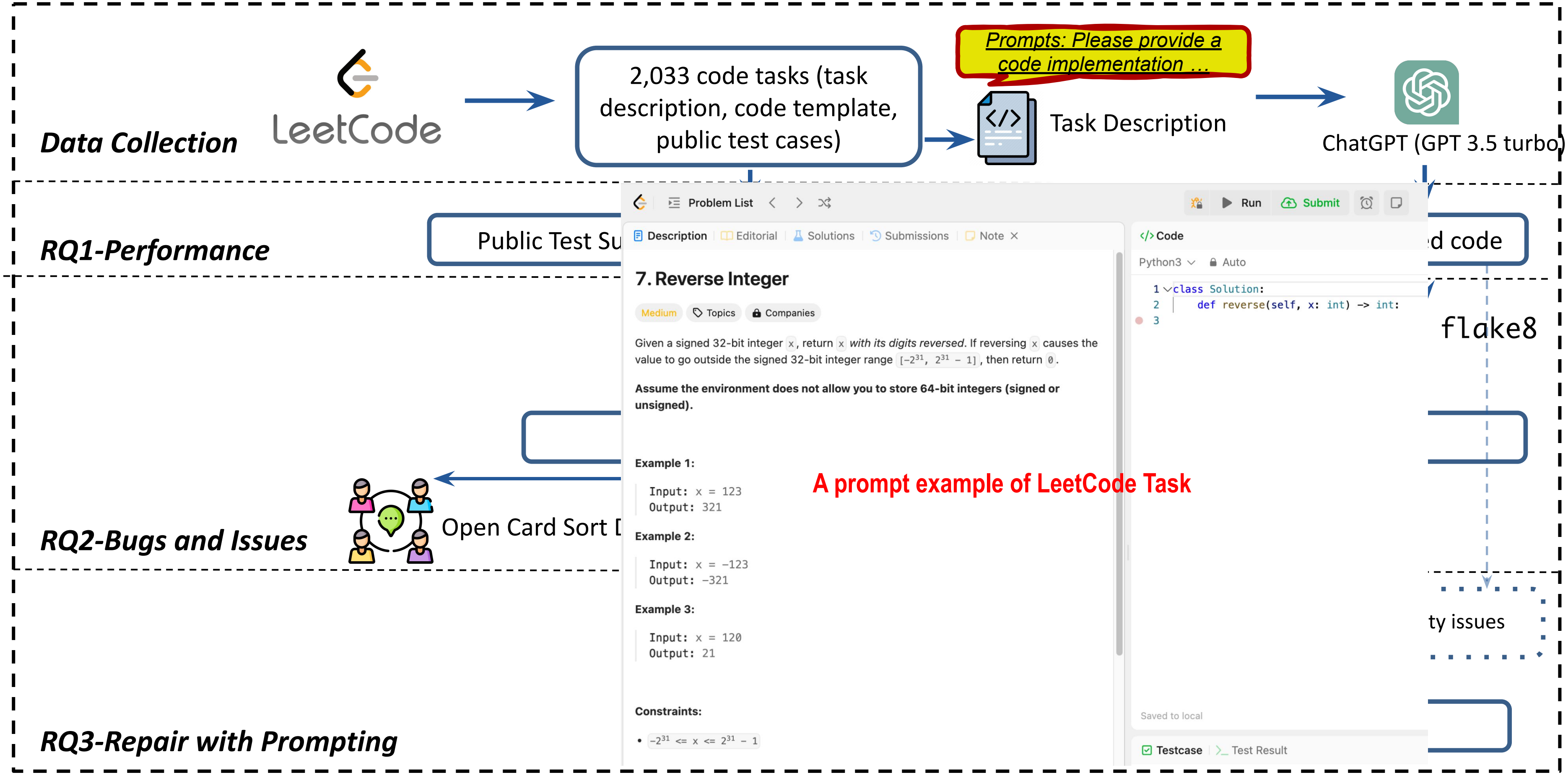
No, no, no! LLM-generated code could also be of low quality or have a bad design. We need to measure it, buddy!



RQ2: Is the code generated by LLMs reliable enough to be applied in the real world? How do they influence the reliability?

Liu, Yue, et al. "Refining ChatGPT-generated code: Characterizing and mitigating code quality issues." *ACM Transactions on Software Engineering and Methodology* (2024).

Part II: Reliability of Code Generated by LLM-based Tools



ChatGPT Can't Always Generate High-Quality Code

	Easy (501)		Medium (1064)		Hard (468)		Pass (2756)	Fail (1310)	Sum
	P	J	P	J	P	J			
Compilation and Runtime Error	7 (1%)	8 (2%)	37 (3%)	32 (3%)	46 (10%)	47 (10%)	0 (0%)	177 (14%)	177 (4%)
Wrong Outputs	47 (9%)	60 (12%)	290 (27%)	260 (24%)	229 (49%)	196 (42%)	0 (0%)	1082 (83%)	1082 (27%)
Code Style and Maintainability	174 (35%)	230 (46%)	431 (41%)	588 (55%)	194 (41%)	313 (67%)	1243 (45%)	687 (52%)	1930 (47%)
Performance and Efficiency	1 (0%)	2 (0%)	20 (2%)	16 (2%)	6 (1%)	6 (1%)	0 (0%)	51 (4%)	51 (1%)

Key Findings

- Code quality issues commonly happen in both code that pass or failed test cases, highlighting the need for characterizing and addressing these concerns alongside the functional correctness.
- Issues in ChatGPT-generated code can be categorized into four categories: Compilation & Runtime Errors, Wrong Outputs, Code Style & Maintainability, Performance & Efficiency
- Wrong Outputs and Code Style & Maintainability issues are the most common challenges faced by the ChatGPT-generated code, while Compilation & Runtime Errors and Performance & Efficiency issues are less prevalent.

```java
class Solution {
    public String makeFancyString(String s) {
        int n = s.length();
        if (n < 3) {
            return s;
        }
        char[] charArr = s.toCharArray();
        for (int i = 2, j = 2; i < n; i++) {
            if (charArr[j - 2] != charArr[i]) {
                charArr[j++] = charArr[i];
            }
        }
        return new String(charArr, 0, j);
    }
}
```

Generated Java Code. Variable "j" is used outside the "for" loop

ChatGPT Can't Always Generate High-Quality Code

	Easy (501)		Medium (1064)		Hard (468)		Pass (2756)	Fail (1310)	Sum
	P	J	P	J	P	J			
Compilation and Runtime Error	7 (1%)	8 (2%)	37 (3%)	32 (3%)	46 (10%)	47 (10%)	0 (0%)	177 (14%)	177 (4%)
Wrong Outputs	47 (9%)	60 (12%)	290 (27%)	260 (24%)	229 (49%)	196 (42%)	0 (0%)	1082 (83%)	1082 (27%)
Code Style and Maintainability	174 (35%)	230 (46%)	431 (41%)	588 (55%)	194 (41%)	313 (67%)	1243 (45%)	687 (52%)	1930 (47%)
Performance and Efficiency	1 (0%)	2 (0%)	20 (2%)	16 (2%)	6 (1%)	6 (1%)	0 (0%)	51 (4%)	51 (1%)

Key Findings

- Code quality issues commonly happen in both code that pass or failed test cases, highlighting the need for characterizing and addressing these concerns alongside the functional correctness.
- Issues in ChatGPT-generated code can be categorized into four categories: Compilation & Runtime Errors, Wrong Outputs, Code Style & Maintainability, Performance & Efficiency
- Wrong Outputs and Code Style & Maintainability issues are the most common challenges faced by the ChatGPT-generated code, while Compilation & Runtime Errors and Performance & Efficiency issues are less prevalent.

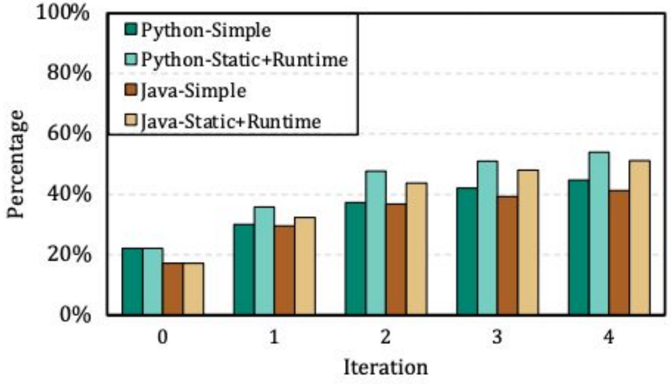
```python
def getMinDistance(self, nums: List[int], target:
    int, start: int) -> int:
    min_diff = float('inf')
    min_index = -1
    for i in range(len(nums)):
        if nums[i] == target:
            diff = abs(i - start)
            if diff < min_diff:
                min_diff = diff
                min_index = i
    return min_diff
```

"min_index" is unused -> smelly code

Repairing Code Quality Issues with Prompting

Prompt Strategies

- Simple feedback (No details)
- Feedback from static analysis and compiler
- Iterative feedback



Iterative Feedback Impact on Producing Code Without Quality Issues

Key Findings

- Prompts with detailed feedback can effectively assist ChatGPT in self-repairing code quality issues, whereas ambiguous feedback may have a negative impact on ChatGPT's performance.
- Iterative repairing proves to be effective, particularly when guided by detailed feedback that incorporates static analysis and runtime errors.

Liu, Yue, et al. "Refining ChatGPT-generated code: Characterizing and mitigating code quality issues." *ACM Transactions on Software Engineering and Methodology* (2024).

Part II: Reliability of Code Generated by LLM-based Tools

- **ChatGPT-generated Code Include Low-quality Issues**: Issues in ChatGPT-generated code can be categorized into four categories: Compilation & Runtime Errors, Wrong Outputs, Code Style & Maintainability, Performance & Efficiency
- **Repairing Code Quality Issues with Prompting is Useful**: Prompts with detailed feedback can effectively assist ChatGPT in self-repairing code quality issues

RQ2: Is the code generated by LLMs reliable enough to be applied in the real world? How do they influence the reliability?

Answer: While LLMs like ChatGPT can generate code when developing software, this code often contains low-quality elements such as bugs or code smells, which can affect overall reliability.
Future work: Enhance LLMs' self-repair capabilities through improved prompting strategies; Establish robust evaluation means to ensure high code quality standards.

Liu, Yue, et al. "Refining ChatGPT-generated code: Characterizing and mitigating code quality issues." *ACM Transactions on Software Engineering and Methodology* (2024).

Part III: Reliability of LLM-based Software Development Applications

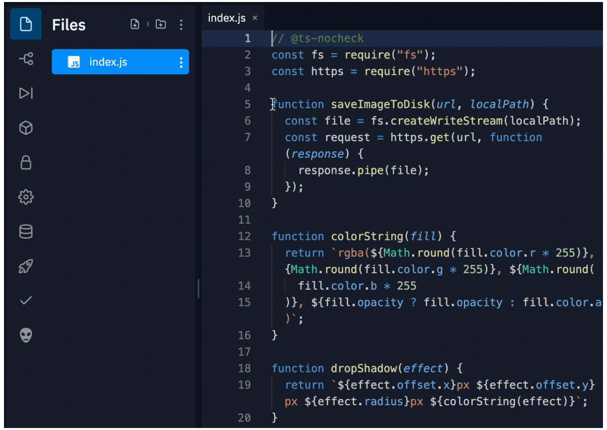
Could I be any more free? These LLM-based software development tools in my IDEs are my Joey. They're my lobster in the coding sea. I don't just use them, I rely on them. They're knocking on productivity's door!

Software Development Environment

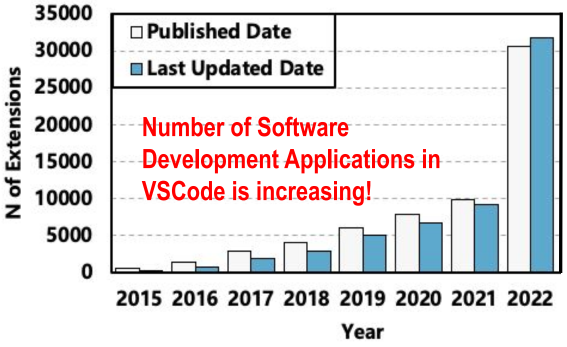
Software Development Applications in IDEs

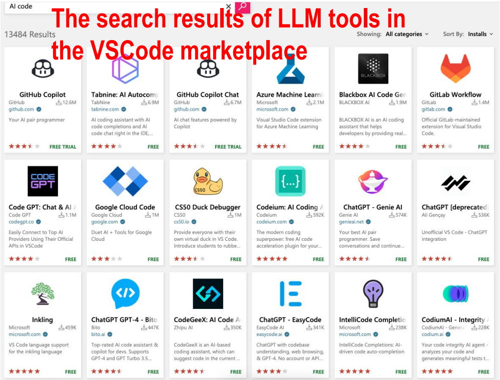
Higher Productivity for Software Developers

- Code Generation
- Code Repair
- Code Translation
- Code Review
- Code Completion
- Code Understanding
- Code Commit Generation
- Program Synthesis
- ……



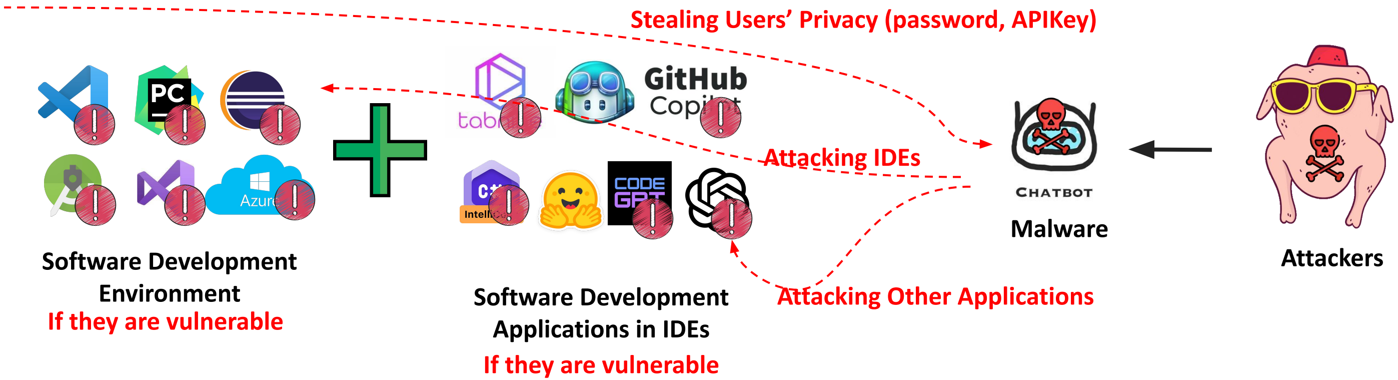

```
// @ts-nocheck
const fs = require("fs");
const https = require("https");

function saveImageToDisk(url, localPath) {
  const file = fs.createWriteStream(localPath);
  const request = https.get(url, function
  (response) {
    response.pipe(file);
  });
}

function colorString(fill) {
  return `rgba(${Math.round(fill.color.r * 255)},
  ${Math.round(fill.color.g * 255)}, ${Math.round(
    fill.color.b * 255
  )}, ${fill.opacity ? fill.opacity : fill.color.a
  )`;
}

function dropShadow(effect) {
  return `${effect.offset.x}px ${effect.offset.y}
  px ${effect.radius}px ${colorString(effect)}`;
}
```





Part III: Reliability of LLM-based Software Development Applications

Could I be any more free? These LLM-based software development applications in my IDEs are my Joey. They're my lobster in the coding sea. I don't just use them, I rely on them. They're knocking on productivity's door!



No, no, no! Y'know, sometimes, you just can't trust completely. We don't know weather IDEs or applications are secure. Hackers could be out there and attack you. It's like when I lost my sandwich, you just never know when it's going to happen!

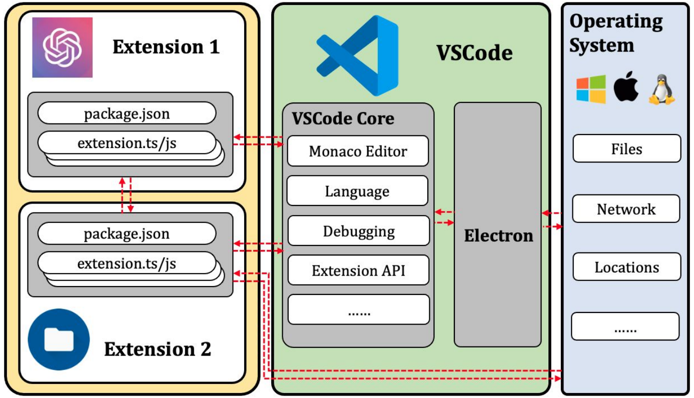
RQ3: Are LLM development tools in our development environment reliable for use? How do they influence the reliability?

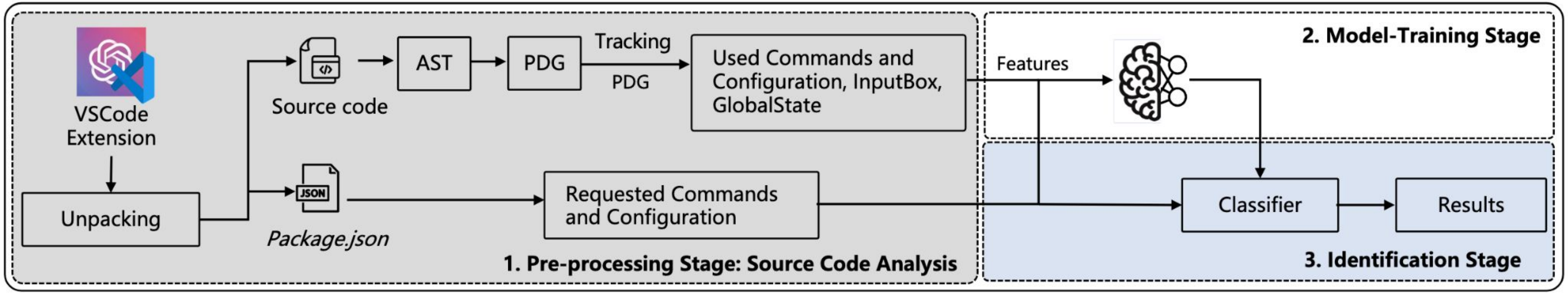
Liu, Yue, et al. "Protect Your Secrets: Understanding and Measuring Data Exposure in VSCode Extensions" *Submitted* (2024).

Development Environment (VSCode)

Key Differences from popular software ecosystem:

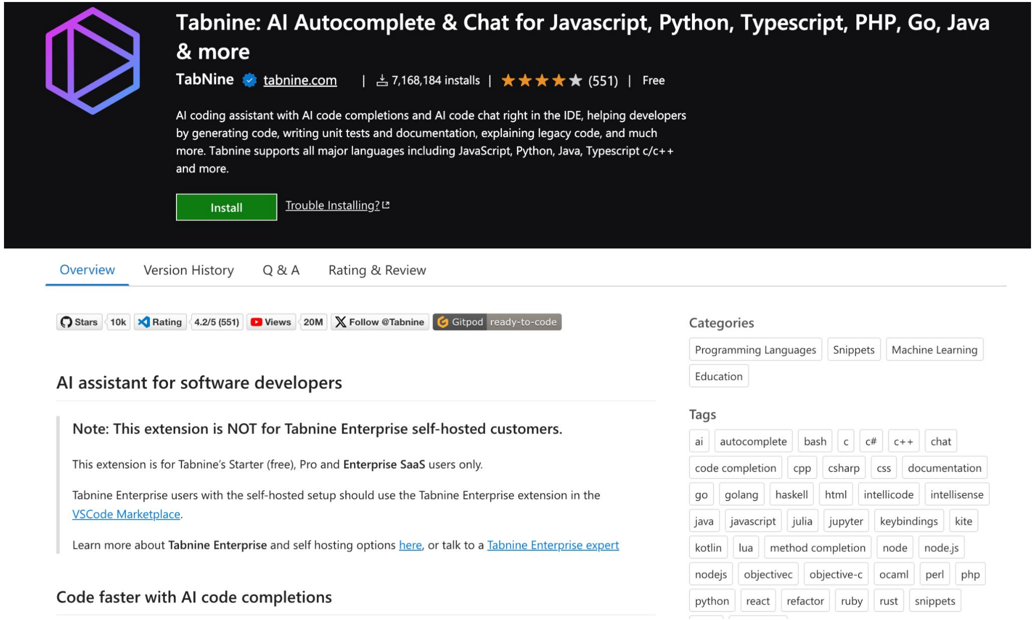
- **No Permission Protocols**: Extensions can access resources or carry out functions without permission granted by the host apps;
- **Event-Driven Activation**: Extension is launched by specific events;
- **Framework Differences**: A set of privileged official APIs

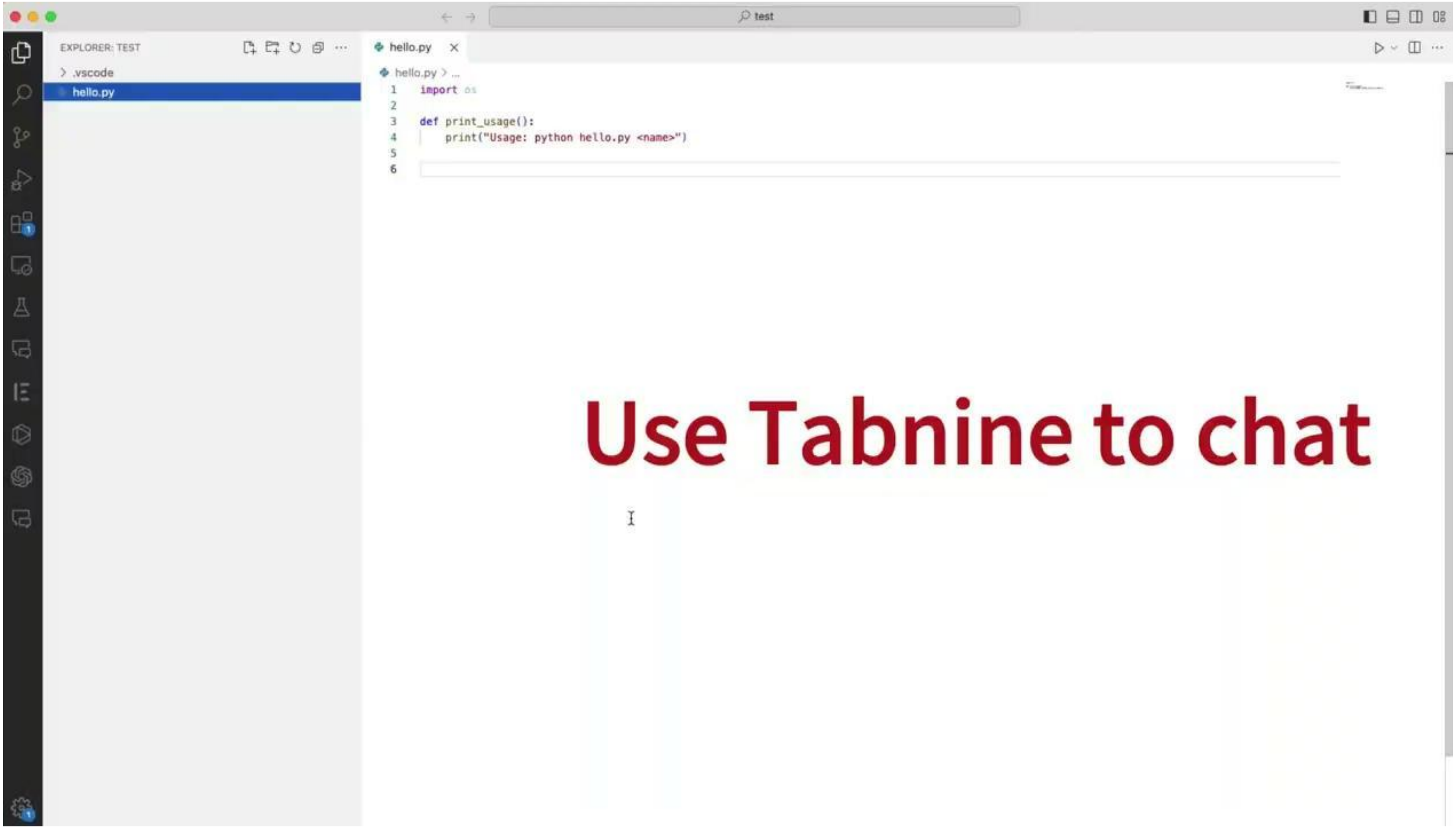




Security Risks for VSCode Extensions

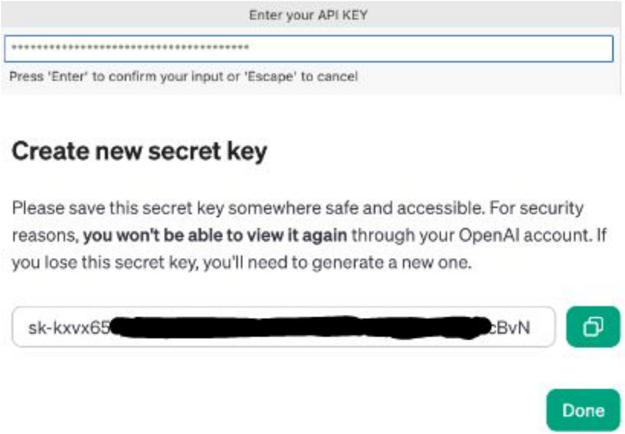
- **Improper Credential Storage**: Despite the design of VSCode extensions to operate in isolation, not all data within an extension is isolated. Attackers can access other extensions' configuration and storage (Tabnine, EasyCodeAI).




test
EXPLORER: TEST
.vscode
hello.py
hello.py
hello.py > ...
import os
def print_usage():
print("Usage: python hello.py <name>")
Use Tabnine to chat

Security Risks for VSCode Extensions

- **Access to In-Extension Sensitive Storage**: Despite the design of VSCode extensions to operate in isolation, not all data within an extension is isolated. Attackers can access other extensions' configuration and storage (Tabnine, EasyCodeAI).
- **Clipboard Snooping**: Clipboard snooping is a security threat that malicious extensions can use to access the clipboard and steal sensitive information that users copy from other sources.



Security Risks for VSCode Extensions

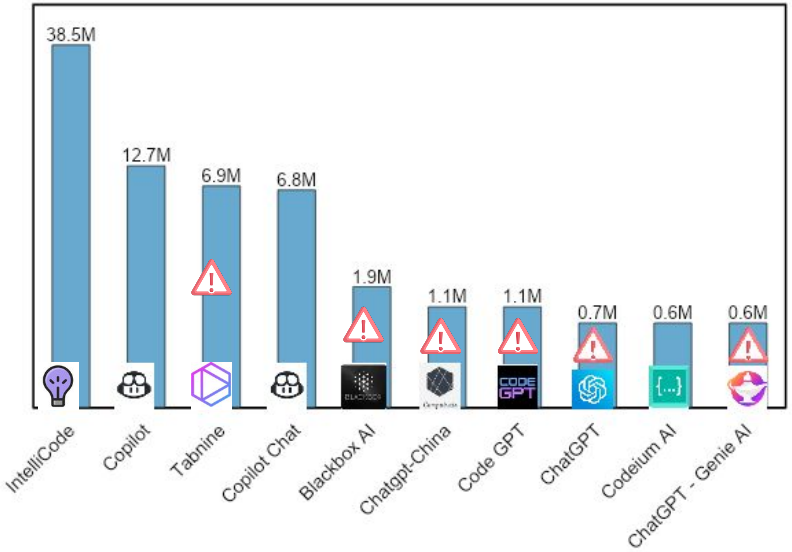
- **Access to In-Extension Sensitive Storage**: Despite the design of VSCode extensions to operate in isolation, not all data within an extension is isolated. Attackers can access and update other extensions' configuration and storage (e.g, Tabnine, EasyCodeAI).
- **Clipboard Access**: Clipboard snooping is a security threat that malicious extensions can use to access the clipboard and steal sensitive information that users copy from other sources (e.g., Chat-GPT).
- **Credential Control**: Extensions can define commands to control various operations, including handling sensitive information. Other extensions can execute these operations using the official API *commands.executeCommand*. (e.g., CodeGPT)

Exposed Type		Items per Exts	# Extensions	Total
Storage Access	GlobalState	1.38	316 (18.0%)	1599
	Requested Configuration	1.43	1205 (9.6%)	
	Used Configuration	1.23	295 (2.7%)	
Clipboard Access	InputBox	1.22	620 (11.5%)	620
Credential Control	Requested Commands	1.65	593 (2.7%)	724
	Used Commands	1.43	458 (2.3%)	

Security Risks for VSCode Extensions

Key Findings

- Out of the extensions analyzed, 2,325 pose a risk of leaking credentials ;
- For LLM-based software development applications, relying more on privacy can lead to more risk. Bad software design can make it difficult to deal with this risk.



Part III: Reliability of LLM-based Software Development Applications

- **Exposure of User Credentials in VSCode Extensions**: Our analysis of 27,261 real-world VSCode extensions revealed that 8.5% (2,325 extensions) are vulnerable to credential-related data leaks. These leaks can occur through various channels, including commands, user inputs, and configurations.

RQ3: Are LLM development applications in our development environment reliable for use? How do they influence the reliability?

Answer: The current state of LLM-based development applications is not sufficiently reliable. They have security flaws that could potentially leak users' private data, such as credential-related information.
Future work: Enhancing the security and reliability of LLM-based development tools is crucial.

Liu, Yue, et al. "Protect Your Secrets: Understanding and Measuring Data Exposure in VSCode Extensions" *Submitted* (2024).

Summary

Overarching RQ: What are the key factors/issues that could impact the reliability of LLM-based software development tools, and how do they influence their reliability?

Reliability & Truthfulness

- 100% **Data duplication and lack of diversity in benchmark datasets**
  - Unreliable benchmark datasets can result in misleading evaluation results
- **Code quality issues such as bugs and code smells in LLM-generated Code**
  - LLM tools that produce low-quality code are not reliable for real-world usages
- **Exposure of User Credentials in software development applications in VSCode**
  - Security flaws undermine the reliability of LLM-based applications

Future Work

- Prompt design for reliable LLM-based software development tools;
- Explore strategies to help the LLM learn from developers' activities in IDEs, with the aim of enhancing both efficiency and productivity;
- Impact of LLM vulnerabilities on LLM-based software development applications;

Could I BE any more excited? Reliable LLM-based software development tools have turned me into the Chan-Chan Man. I'm as free as a bird!

More time at home? Now you'll have more time to help me organize the spice rack and perfect our lasagna recipe! This is the best news ever!

Oh…my…GAWD! Y'know, this reliable LLM thing? It's gonna put Chandler Bing right out of a J-O-B-B-Y job! No more coding for him!

Sincere thanks to everyone who supported and helped me throughout my 5-year PhD journey!