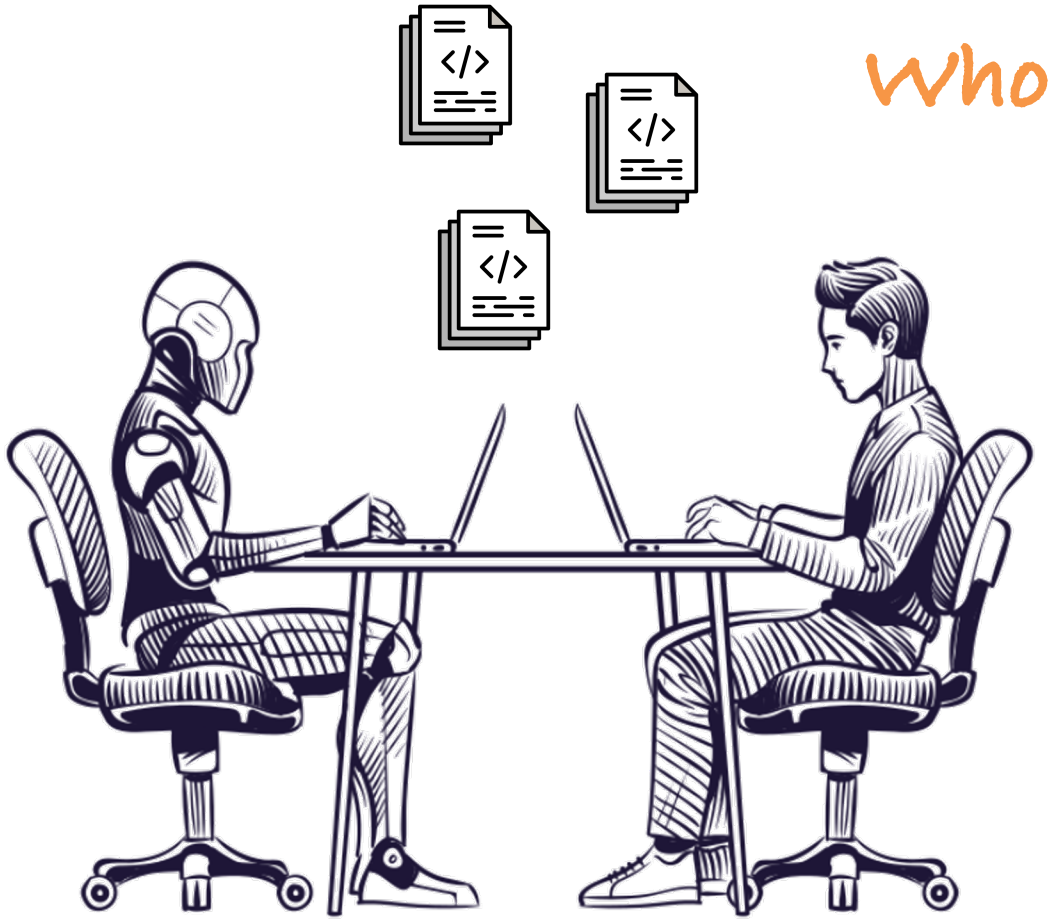
# Between Lines of Code: Unraveling the Distinct Patterns of Machine and Human Programmers

**Yuling Shi**[1], **Hongyu Zhang**[2], **Chengcheng Wan**[3], **Xiaodong Gu**[1*]

[1]Shanghai Jiao Tong University, [2]Chongqing University, [3]East China Normal University



# AI-Generated Code Detection



Who wrote these codes?

# Why Distinguish AI-Generated Code?

- **Plagiarism & Authenticity Concerns:**

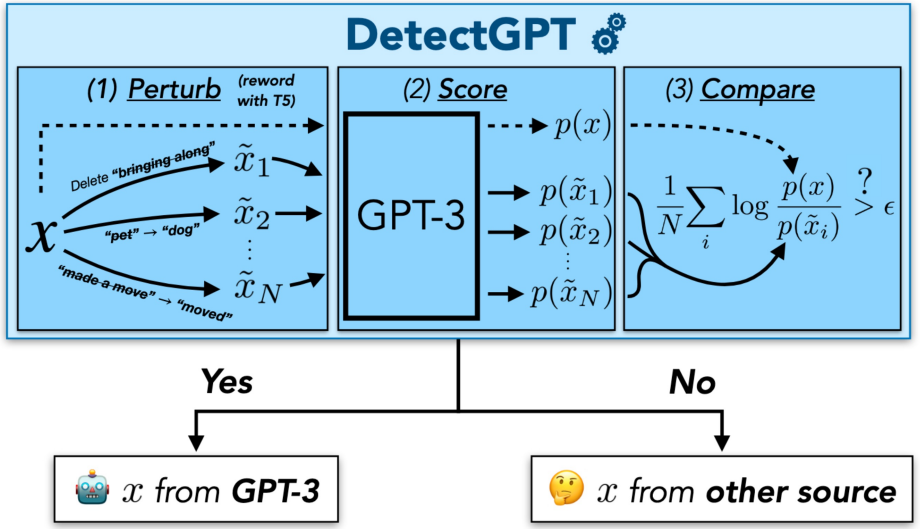
  Unclear Code Source; Challenges Academic & Interview Fairness

- **Potential Flaws & Quality Issues:**

  Requires Strict Review & Testing; Not Guaranteed Optimal or Fully Compliant


# Previous Approach

- DetectGPT analyzes the likelihood score discrepancies between original and perturbed texts.




- **Does not identify the unique patterns of AI-generated code.**

(Mitchell et al., 2023)

# Previous Approach

- SWEET watermarks the code generation model to embed special features.

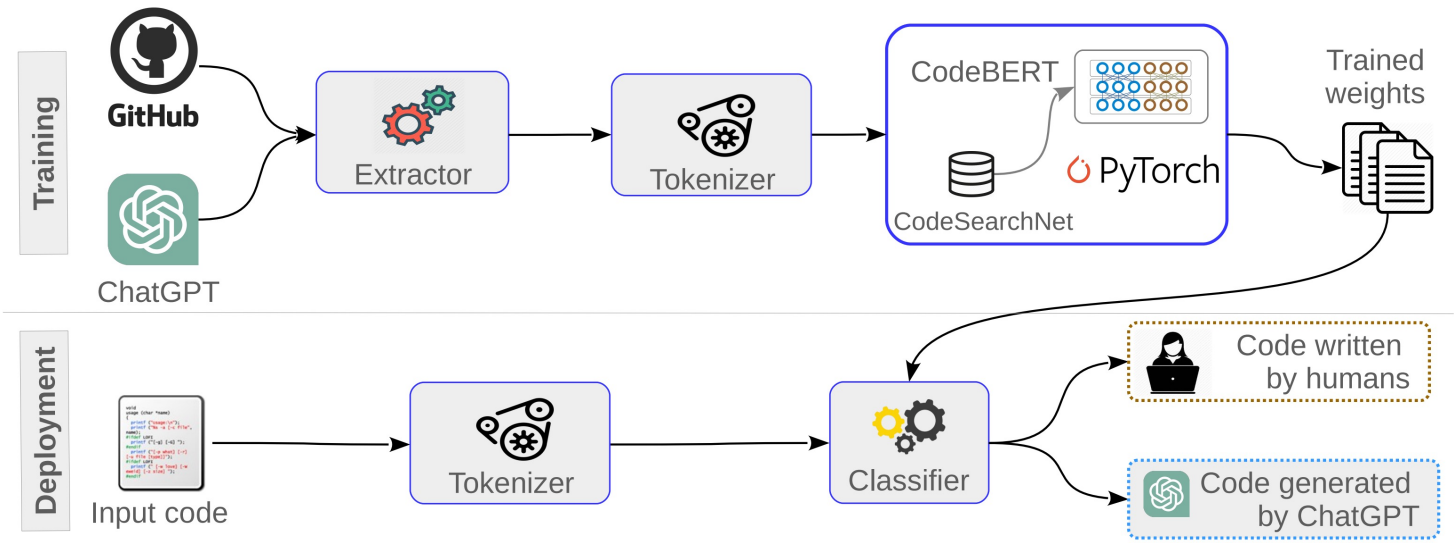
```
    centerpoint = sum(numbers) / len(numbers)
    distances = []
    for number in numbers:
        distance = abs(number - centerpoint)
        distances.append(distance)
    m_a_d = sum(distances) / len(distances)
    return m_a_d
```

- **Only applicable to detect codes from watermarked model.**

(Lee et al., 2024)

# Previous Approach

- GPTSniffer trains a supervised CodeBERT model as the classifier.




- **Requires continuous data collection and training.**

(Nguyen et al., 2024)

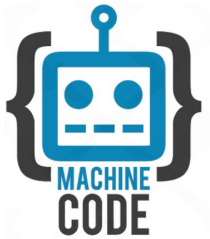
# Our Contribution

- A **comprehensive empirical analysis** of AI and human's codes, focusing on **diversity, conciseness, and naturalness**.
- A **novel zero-shot method for detecting** AI-generated code

# Empirical Analysis Design

**What are the differences between AI and human's codes?**

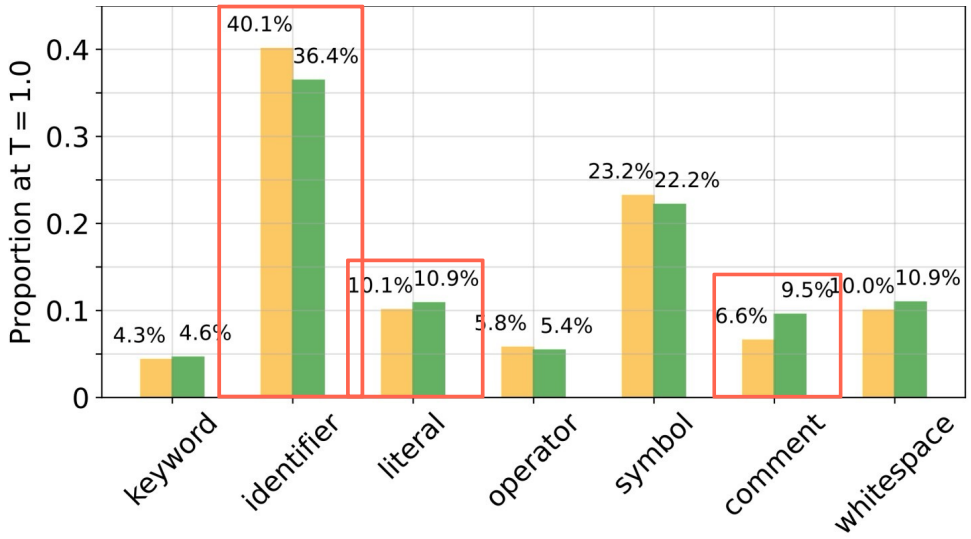
| Dimension | Metrics |
| --- | --- |
| Lexical Diversity | Vocabulary for different parts of code |
| Conciseness | Number of tokens and lines of code |
| Naturalness | Token likelihood and rank |

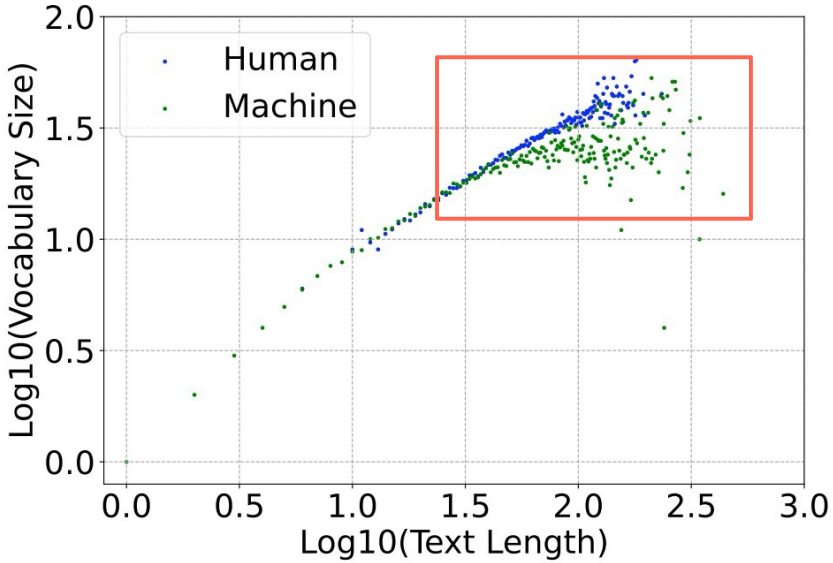


# Dataset Collection

- **Source:** 10,000 Python functions sampled from CodeSearchNet.
- **Model:** CodeLlama (7B) as the code generation model.
- **Prompt:** Function signatures and comments as prompts.

# Results on Lexical Diversity

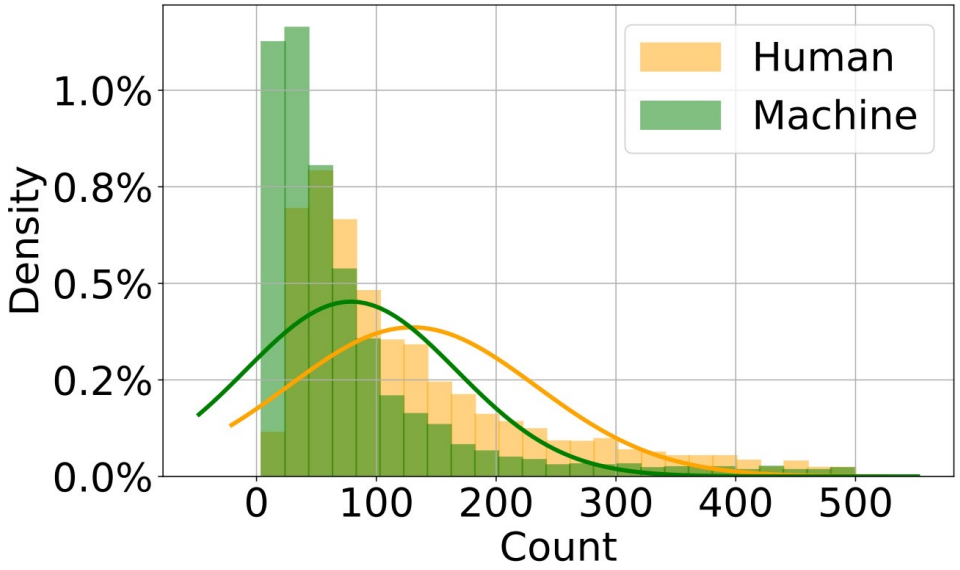


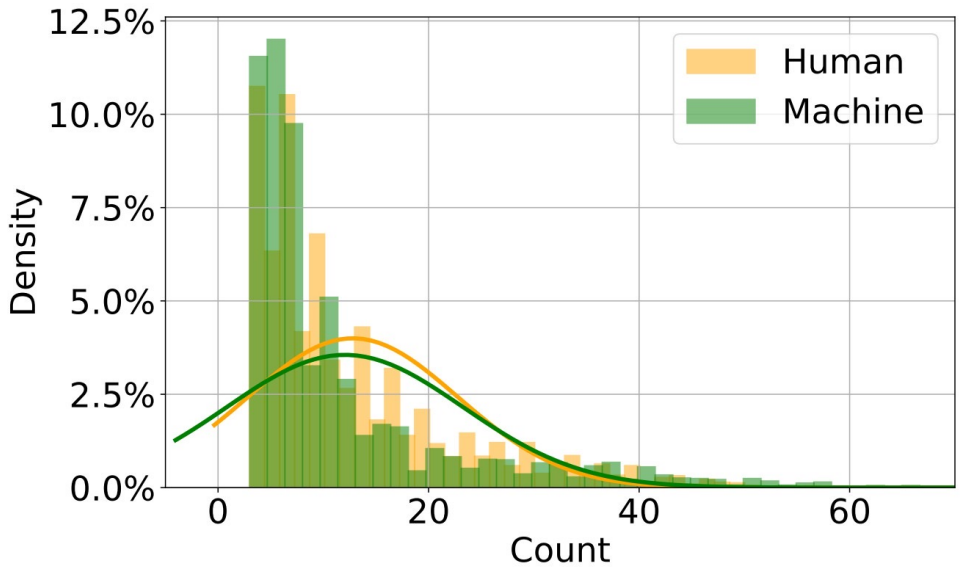




**Finding 1:** AI's code **use fewer identifiers, more literals** and **comments.**

**Finding 2:** AI's code shows a preference for a **limited set of frequently-used tokens**.

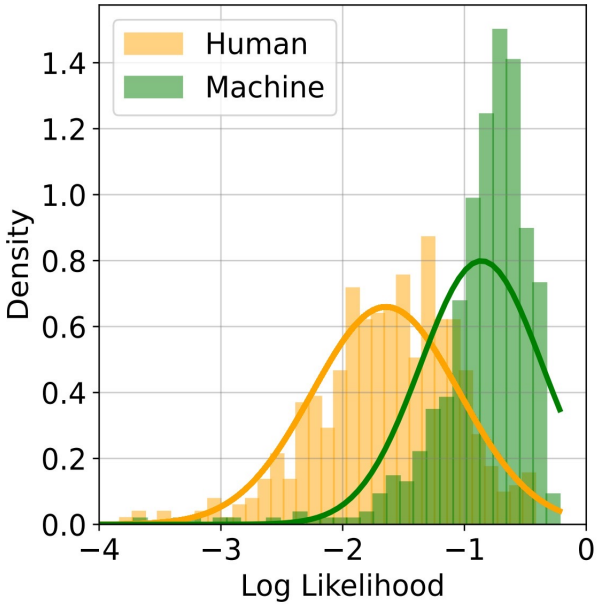
# Results on Conciseness

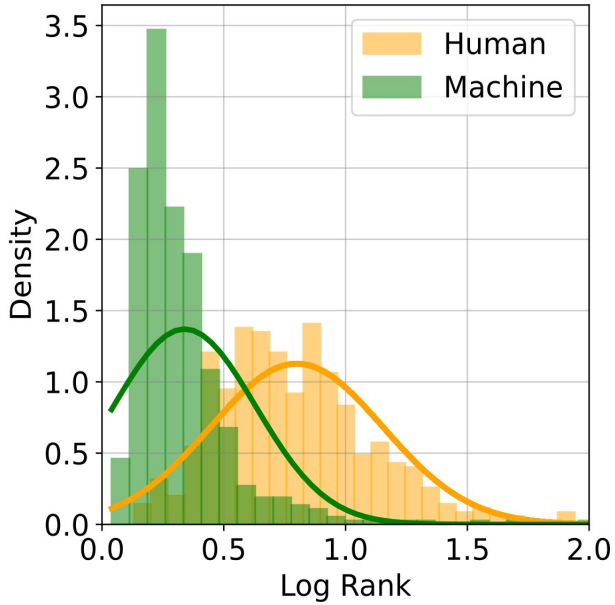



> **Finding 3:** LLM's code is **generally more concise, with fewer tokens and lines**, while human code is more varied.

# Results on Naturalness



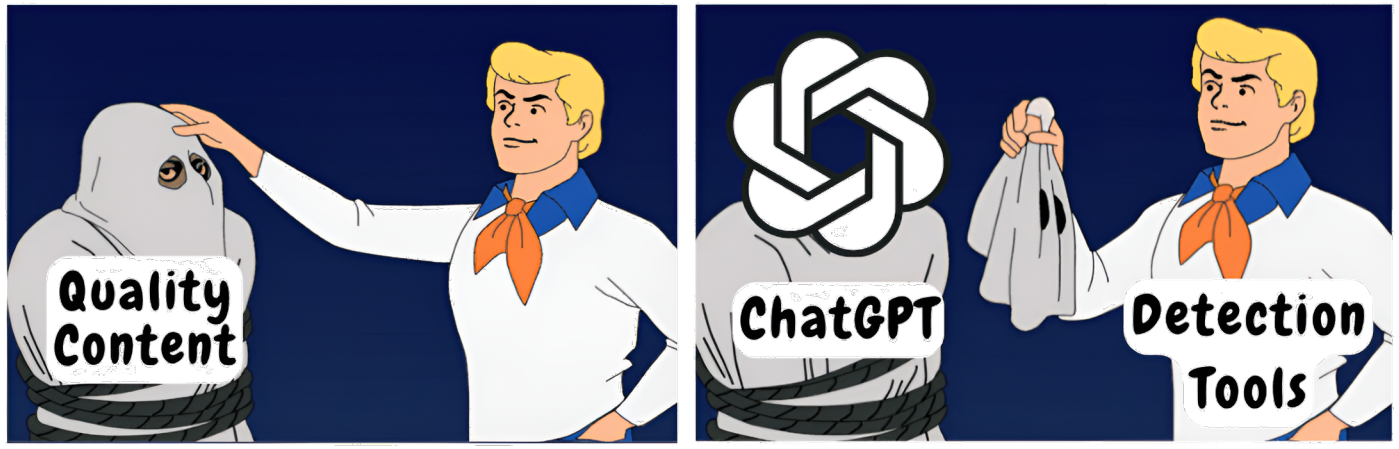




| Category | Log Likelihood | | |
|---|---|---|---|
| | Machine | Human | Δ |
| keyword | -1.701 | -2.128 | 0.428 |
| identifier | -0.459 | -0.874 | 0.415 |
| literal | -0.506 | -1.364 | 0.858 |
| operator | -0.938 | -1.835 | 0.897 |
| symbol | -0.868 | -1.639 | 0.771 |
| comment | -1.503 | -3.028 | 1.525 |
| whitespace | -1.131 | -2.740 | **1.609** |
| ALL | -0.827 | -1.658 | 0.831 |

**Finding 4:** LLM's code **exhibits higher naturalness** compared to human code.

**Finding 5:** The most significant naturalness differences are in **comments and stylistic tokens like whitespaces.**

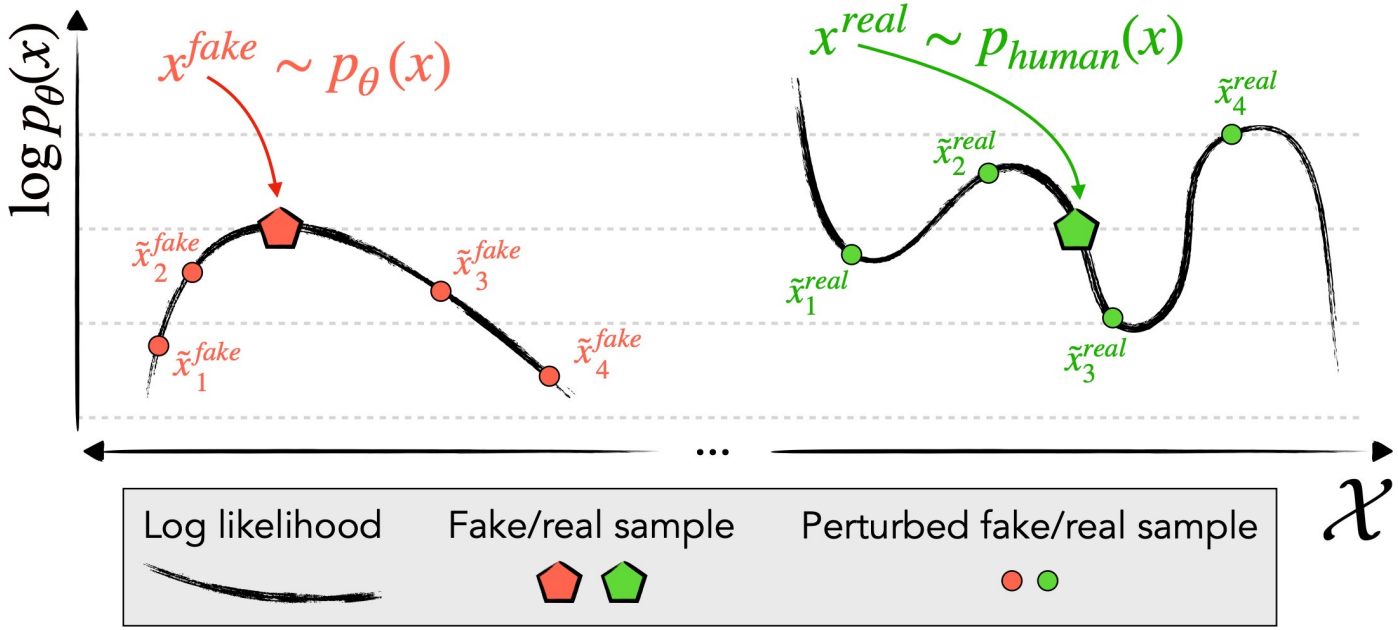
# DetectCodeGPT: A Zero-shot Code Detection Approach




No training data required

# Methodology

- **After perturbations:**
  - AI-generated code will exhibit **significant changes** in naturalness
  - Human-written code **exhibits limited variation** in naturalness.



# Stylized Perturbation

- **Before Perturbation**:

```
def calculate_tax(income,tax_rate):
    if income>10000:
        return income*0.2
    else:
        return income*0.1
```

- **After Perturbation**:

```
def calculate_tax(income, tax_rate):
    🎯
    if income 🎯 > 🎯 10000:
        return income*0.2
    🎯
    else:
        return income🎯*🎯0.1
```

Randomly Perturb on Spaces and Newlines

# Stylized Perturbation

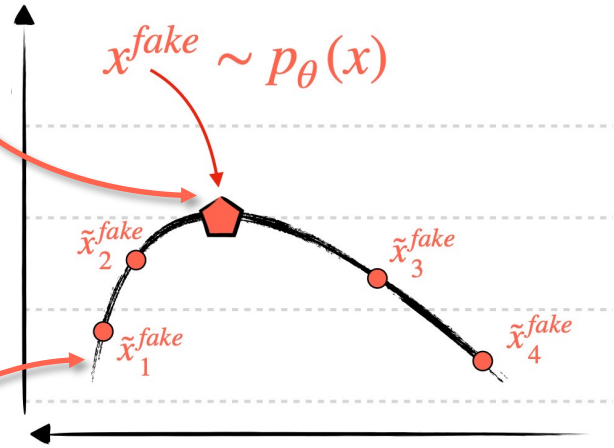
- **Before Perturbation**:

```
def calculate_tax(income,tax_rate):
    if income>10000:
        return income*0.2
    else:
        return income*0.1
```

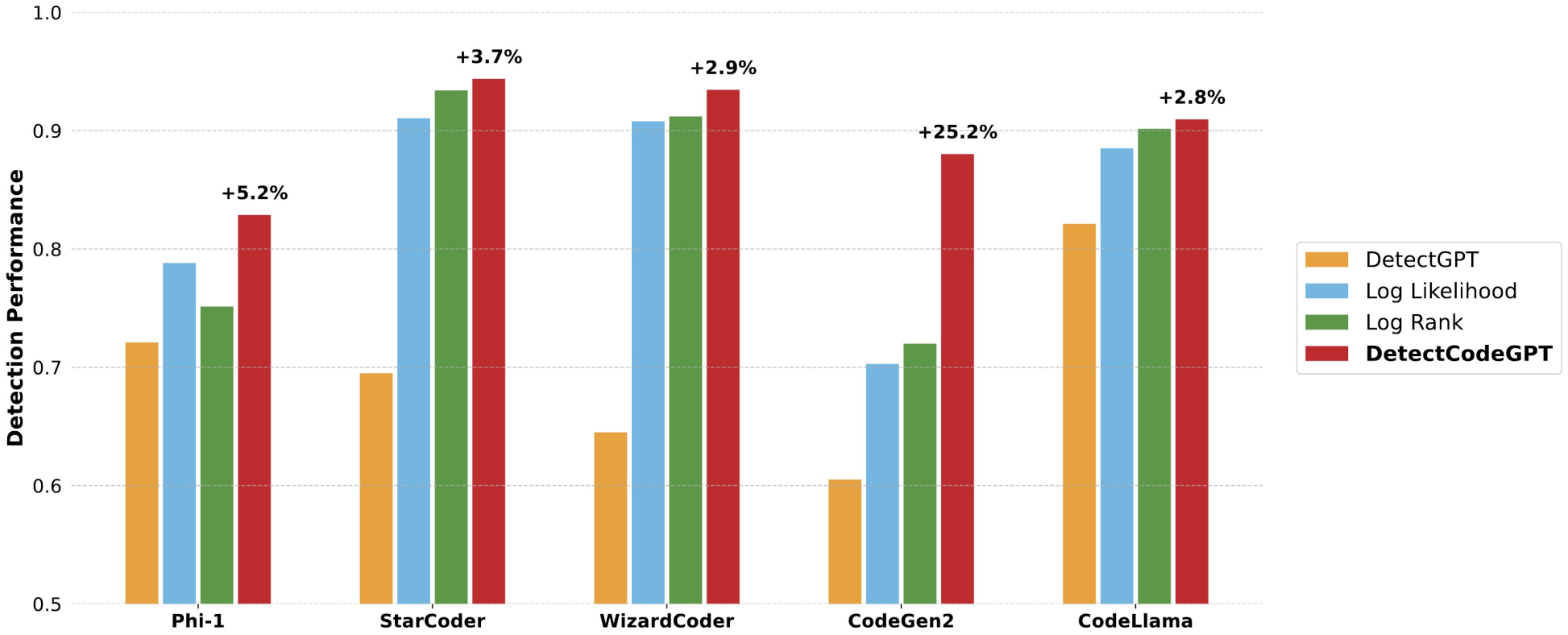
- **After Perturbation**:

```
def calculate_tax(income, tax_rate):
    🎯
    if income 🎯 > 🎯 10000:
        return income*0.2
    🎯
    else:
        return income🎯*🎯0.1
```

Randomly Perturb on Spaces and Newlines



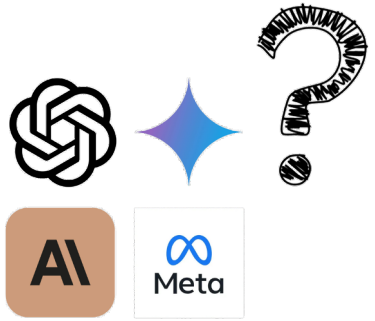
# Detection Performance




**~7.6%** higher then pervious SOTA

# Cross-Model Detection

**In real world scenario, the source LLM is usually unknown.**


> **We can use one model to detect contents from another.**

# Case Study

```
import os
import re

if not os.path.isdir(path):
        raise ValueError('%s is not a directory' % path)
↵
if not isinstance(include, (list, tuple)):
        include = [include]
if not isinstance(exclude, (list, tuple)):
        exclude = [exclude]
↵
if not show_all:
        exclude.append(r'\.pyc$')

...
```

**Truth:** 🤖

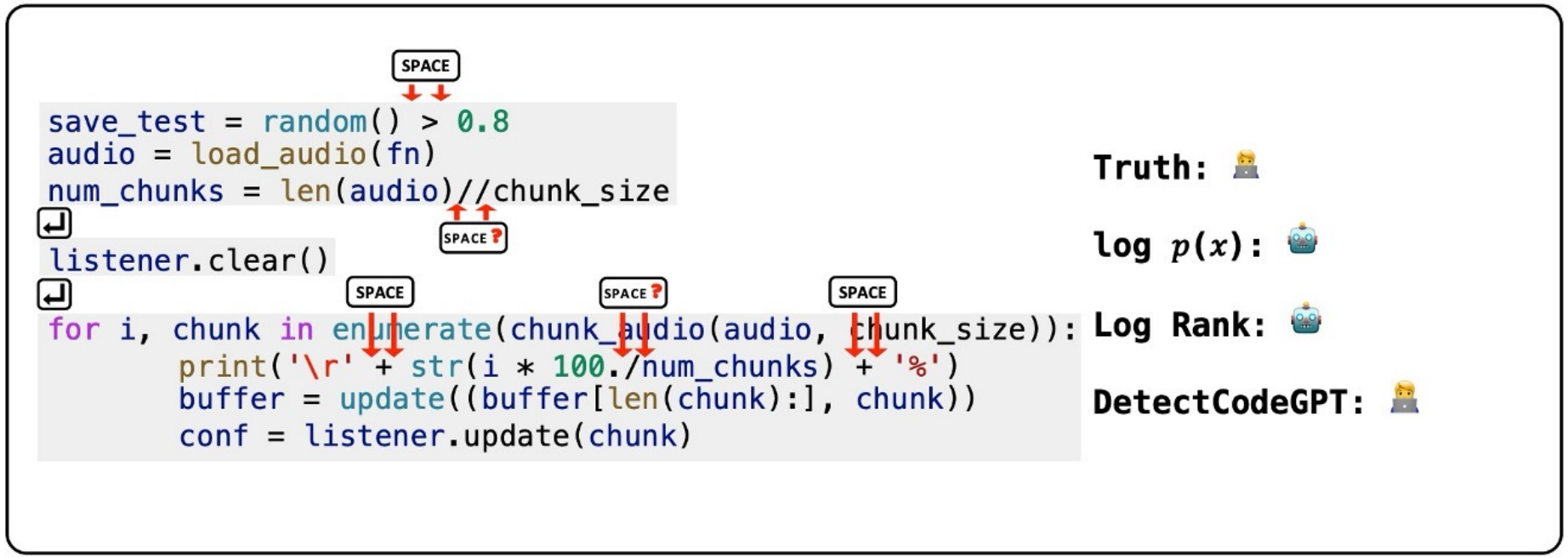
**log $p(x)$:** 👨‍💻

**Log Rank:** 👨‍💻

**DetectCodeGPT:** 🤖

> **Illustrating AI's structured coding patterns**

# Case Study




> **Revealing human's randomness in coding.**

# Future Directions

- **Language:** Investigate across **other programming languages**.
- **Performance:** Explore **more features** in AI's codes.
- **Attack:** Design **adversarial attacks** to evade detection.
- **Defense:** Propose **defense strategies** against these attacks.

# Thank you for your attention

Our data and code are available at https://github.com/YerbaPage/DetectCodeGPT

Feel free to reach out for any question or collaboration at yuling.shi@sjtu.edu.cn