Summary of "Whodunit: Classifying Code as Human Authored or GPT-4 generated- A case study on CodeChef problems"

Kevin Jie

What Problem Is Being Solved?

Educators worry that students may submit AI-generated code as their own, and existing plagiarism tools struggle because AI code often is not textually similar to student work. The paper asks whether code stylometry and ML can reliably tell GPT-4-generated Python solutions from human ones?

What Are the New Ideas?

The authors build a stylometry-based classifier (XGBoost) using 140 layout, lexical, syntactic, and complexity features (e.g., average line length, AST node stats, cyclomatic complexity) to distinguish GPT-4 code from human solutions. They create a balanced dataset from CodeChef: 399 problems, 1,596 solutions (half human and half GPT-4), rebinned into easy/medium/hard for fair comparisons.

Their system design of the study comprehensively compares:

1. overall performance vs. multiple baselines
2. performance without "gameable" features (like whitespace/empty lines)
3. training only on correct solutions for semantics independence
4. performance across difficulty levels for complexity independence.

They use SHAP to explain model decisions and reasoning. The evaluation results show that overall F1 and AUC-ROC of 0.91; even without gameable features, F1/AUC-ROC stay at 0.89; performance is similar for easy/medium and only slightly lower on hard tasks.

Positive Points:

The paper uses a robust methodology with equal numbers of human and AI solutions across different levels of difficulty problems. It employs 140 stylometry and complexity features, and GroupKFold by problem prevents data leakage. Baselines such as random and n-gram models are tested for fair comparison. The approach remains effective even when restricted to non-gameable features. Finally, the authors use SHAP for both global and local feature explanations, which improves interpretability and builds human trust in the detection process.

Negative Points:

The study is platform-dependent since it uses only CodeChef, and coding practices may differ elsewhere. It samples the most popular 100 problems per difficulty with Python solutions, but may not capture all coding styles. The dataset also includes only GPT-4 for AI-generated code, so results may not generalize to other assistants. Finally, the correctness of AI code was checked only against CodeChef's public test cases, not private ones.

Future Work

Future work could involve large-scale, real-world validation by testing on actual course submissions with consent and anonymization. Classifier judgments could then be compared against instructor-labelled ground truth. Since the current study uses only Python, expanding to languages like Java, C++, and JavaScript would test whether the stylometric signals remain consistent.

Rating

4 / 5: The paper is rigorous and has an interesting topic

In-Class Discussion

Discussion Point 1: How to use this research in practice?

The original motivation of this work is plagiarism detection, but the model itself will need constant updates as both AI and human coding practices evolve. A key concern is minimizing false positives. Accuracy alone is not enough because we need interpretability. The algorithm here is explainable, but how interpretable is it really, and how might it be improved? With humans, responsibility is clear; with machines, accountability is murkier. At present, we cannot even prove code is human-generated. This creates a cat-and-mouse game

where advances in AI require stronger detection and explanation tools.

Discussion Point 2: How interpretable is the model really? How can we make it better?

This connects to a deeper question in education: what are we actually teaching students, and what skills do we want them to develop? If the future of work involves AI assistants, then teaching understanding may matter more than teaching code generation. Writing code might not be the central skill anymore. In the past, people believed you were not skilled if you did not know assembly, but that view has since changed.

Discussion Point 3: What if we try to circumvent these systems?

As the saying goes, we shape our tools and then our tools shape us. With AI coders, we are already seeing this shift. We are becoming more code appraisers than code writers.

Reference

[1] Oseremen Joy Idialu, Mathews, N.S., Rungroj Maipradit, Atlee, J.M. and Nagappan, M. 2024. Whodunit: Classifying Code as Human Authored or GPT-4 Generated - A case study on CodeChef problems. *MSR '24: 21st International Conference on Mining Software Repositories*. (Jul. 2024), 394–406. DOI:https://doi.org/10.1145/3643991.3644926.