HumanVsGenAI: AI Code Detection Performance in Educational Programming Tasks using DetectCodeGPT

Aagat Pokhrel, Chandra Raskoti, Chong Shen, and Iftekharul Islam

Abstract— **The widespread adoption of AI coding tools in educational settings has created new challenges in assessing student work and maintaining academic integrity. This study aims to evaluate the effectiveness of AI code detection systems, specifically DetectCodeGPT, in distinguishing between AI-generated and human-written solutions in programming assignments. Through systematic analysis of detection performance across different types of programming problems, we seek to understand patterns that can inform better assignment design. Our methodology combines collection of paired AI and human solutions with rigorous detection analysis. The expected outcomes include guidelines for designing AI-resistant programming assignments and insights into detection system effectiveness. The project follows a structured 9-week plan, encompassing data collection, analysis, and documentation phases.**

I. Introduction

AI coding assistants such as ChatGPT and GitHub Copilot are transforming programming education by enabling students to approach programming tasks more efficiently. However, this convenience introduces challenges in academic integrity, as AI-generated code becomes increasingly indistinguishable from human-written solutions. Traditional plagiarism detectors, designed to identify human authorship patterns, are inadequate for the nuances of AI-generated content, leading to potential academic misconduct. This growing concern underscores the need for reliable methods to distinguish between AI-generated and human-written code in educational contexts.

Recent advances in AI detection systems, particularly DetectCodeGPT [1], show promise in identifying machine-generated code by analyzing stylistic patterns such as whitespace, indentation, and structural choices. These approaches leverage the subtle differences in writing styles between human programmers and AI-generated solutions. Despite this progress, existing research predominantly evaluates these models on general-purpose datasets, overlooking the unique challenges posed by educational programming tasks, which often require a more nuanced understanding of problem-solving strategies and logical reasoning. Furthermore, the influence of problem complexity on detection accuracy remains underexplored, leaving gaps in our understanding of how AI detectors perform across educational scenarios.

This study bridges these gaps by systematically evaluating DetectCodeGPT's performance on educational programming tasks categorized by complexity levels. By analyzing detection patterns and misclassifications, the study provides empirical insights into the strengths and limitations of current AI code detection methods in academic settings. This approach not only advances the technical understanding of AI-generated code detection but also addresses practical challenges faced by educators in assessing student learning and maintaining academic integrity.

Through this project, we aim to address the following research questions:

- How effective is DetectCodeGPT in detecting AI-generated solutions across educational programming tasks?
- How does problem complexity influence detection accuracy?
- What patterns in misclassifications can inform improvements in detection models and assignment design?

Our team combines experiences in software engineering, AI systems, and educational technology, positioning us well to conduct this investigation.

II. Research Value

A. Research Need and Proposed Results

Our research primarily serves computer science educators who face increasing challenges in verifying student work authenticity. These practitioners require reliable methods to distinguish between AI-generated and human-written code in assignment submissions. The immediate value lies in providing empirically-validated strategies for creating programming assignments that better support authentic assessment.

Educational researchers studying AI's impact on programming education form our secondary audience. For these users, our work will provide valuable data on detection patterns across different problem types and complexity levels. This contribution advances the understanding of how AI detection systems perform in educational contexts.

By addressing these needs, our study aims to achieve the following results from the user's perspective:

- Insights into how problem complexity influences detection accuracy.
- Empirical evidence to improve AI detection models.

B. Measures of Success

Success will be measured through detection accuracy across different complexity levels, along with detailed analysis of false positive and false negative rates, as well as confidence scores. Additionally, the study will examine misclassification patterns to identify factors influencing detection

Aagat Pokhrel, Chandra Raskoti, Chong Shen, and Iftekharul Islam are with Min H. Kao Department of Electrical Engineering and Computer Science at University of Tennessee, Knoxville, TN, USA {apokhre2,craskoti,cshen8,mislam73}@vols.utk.edu

outcomes, providing valuable insights into the reliability and limitations of DetectCodeGPT in educational contexts.

III. Method

Our methodology centers on a systematic evaluation of DetectCodeGPT's performance in an educational context. We will select at least 20 programming problems that represent typical assignments across different difficulty levels and concept areas. For each problem, we will collect both AI-generated solutions (using tools like ChatGPT and GitHub Copilot) and human-written solutions.

The analysis phase involves:

- Running DetectCodeGPT on our collected solutions
- Analyzing detection accuracy across different problem types
- Identifying factors that influence detection reliability
- Documenting patterns in successfully and unsuccessfully detected cases

To validate our results, we will examine failure cases in detail and analyze how different problem characteristics affect detection performance. This will inform our recommendations for assignment design.

IV. Schedule and Project Management

Our project follows a structured 9-week timeline, beginning with technical setup and followed by systematic data collection, analysis, and documentation. Table I provides an overview of the key milestones and objectives for each phase.

A. Risk Management

Key risks include dataset collection delays, technical issues with DetectCodeGPT, and detection accuracy falling below expectations. To mitigate these, the following strategies will be followed:

- **Dataset Collection Delays:** Although human solutions are readily available on platforms like GitHub and LeetCode, the team will focus on a manageable set of well-defined problems and write their own solutions when necessary to ensure timely data collection.

TABLE I
PROJECT SCHEDULE AND MILESTONES

Week	Dates	Objectives	Core Tasks
1	2/24 - 2/28	Initial Setup	Install Ubuntu and requirements
2	3/3 - 3/7	DetectCodeGPT Setup	Replicate GitHub instructions
3	3/10 - 3/14	Problem Curation	Select and document problems
4	3/24 - 3/28	Solution Collection	Generate AI and human solutions
5	3/31 - 4/4	Initial Analysis	Run detection system
6	4/7 - 4/11	Detailed Analysis	Compare performance metrics
7	4/14 - 4/18	Pattern Recognition	Analyze misclassifications
8	4/21 - 4/25	Documentation	Draft research report
9	4/28 - 5/6	Final Delivery	Complete documentation

- **Technical Issues:** Early technical setup and testing will be conducted to identify potential issues. If necessary, alternative detection methods will be explored.
- **Detection Accuracy:** If detection accuracy falls below expectations, the team will document the limitations and discuss potential improvements for future work.
- **Timeline Delays:** Tasks will be redistributed among team members as needed to maintain the project schedule.

B. Resources

Our resource requirements include access to AI coding assistants (ChatGPT, GitHub Copilot), computing resources for running DetectCodeGPT, and storage for our code samples and analysis results. Two of team members have the necessary development environments for the project.

V. Team

Our team organization is designed to leverage individual strengths while ensuring clear accountability for project deliverables. As shown in Table II, each team member has specific primary responsibilities while maintaining flexibility to support other areas as needed.

TABLE II
TEAM ROLES AND RESPONSIBILITIES

Role	Member	Key Responsibilities
Project Coordinator	Iftekharul	Track progress, ensure deadlines
Dataset Curators	Iftekharul, Chong	Problem selection, solution collection
Technical Leads	Aagat, Chandra	Implementation, system operation
Analysis Lead	Aagat	Results interpretation, visualization
Documentation	Team effort	Report writing, presentation

Aagat brings experience in Natural Language Processing and has worked on projects utilizing Large Language Models, providing valuable expertise for our technical implementation. Chandra has worked on various classification tasks in NLP and Language Models, which will help in understanding and adapting the detection system. Chong's background in ML models and data mining algorithms will support our data collection and analysis efforts. Iftekharul's experience with ML models and programming task curation makes him well-suited for coordinating the project and managing the dataset collection.

While none of us has direct experience with AI code detection systems, our combined background in related technologies positions us to effectively tackle this challenge. Regular meetings will be held twice weekly to ensure coordination and address challenges promptly. The project will follow an agile development approach with two-week sprints, allowing us to adapt our methods based on early findings and challenges.

References

[1] Y. Shi, H. Zhang, C. Wan, and X. Gu, "Between lines of code: Unraveling the distinct patterns of machine and human programmers," *arXiv preprint arXiv:2401.06461*, 2024.