DOI: 10.2478/sbeef-2025-0010

LLM-BASED DISHONESTY AND EXCESSIVE COLLABORATION DETECTION IN CYBERSECURITY EDUCATION

AUREL-DRAGOȘ HOFNĂR
BABEȘ-BOLYAI University, Cluj-Napoca, Romania, Faculty of Mathematics and Computer Science,
No. 1 Mihail Kogălniceanu Street, RO-400084 Cluj-Napoca, ROMANIA,
E-mail: aurel-dragos@hofnar.ro

Abstract. ***Artificial intelligence long-term memory models are able to provide answers to a wide range of questions from a wide range of domains and to some extent, is able to detect excessive collaboration in the cybersecurity education field, making use of metadata (logs). There are some key differences between the models and the human factor with regards to completeness and correctness of the results. The aim of this paper is to provide an overview of the current state of the art of artificial intelligence and its use in detecting excessive collaboration in the cybersecurity education field.***

Keywords: Academic integrity, Machine learning in education, Cybersecurity, Behavioral pattern recognition, Log data analysis.

1. INTRODUCTION

With the advancements of artificial intelligence (AI) in the form of large language models (LLM) it has become increasingly harder for students to resist the temptation of using these new technologies as shortcuts towards solving exercises that would help them better understand a problem. It has been observed that students use and to some degree abuse AI [1], [2] even when they are expressly told to not do it. Furthermore, it is a well-known fact that students sometimes also ignore directions that forbid them to collaborate between each other [3]. Because technology is always a double-edged sword, this work targets towards making use of AI and LLM to find possible cheaters for the case in which students collaborate with each other for solving a cybersecurity exercise.

Since most assignments in the traditional education system require students to solve exercises and since in computer science related education programs these exercises produce logs, this work aims at using the logs to query state of the art LLMs to find possible cheaters.

Therefore, this work contributes towards detecting cheating in cybersecurity education using LLMs by measuring different state of the art LLMs ability of detecting possible cheating attempts using a well-engineered prompt and access logs recorded during the exercise solving phase by the students.

2. RELATED WORK

Since the domain of AI and LLMs is fairly new (at least in comparison with other domains within computer science) there is only a yet fairly limited understanding and exploration of this subject. Nonetheless, the recently introduced challenges with regards to maintaining academic integrity have been studied by Zhang et al. [4] who emphasize that although there are clear benefits to LLMs and AI in general, the risk of academic dishonesty also increases proportionally. Furthermore, Beverly et al. [5] studied the impact that AI tools like LLMs have in cybersecurity education and emphasize that exercises should be redesigned to address the possible misuse of these tools. Zou et al. [6] surveyed techniques to recognize AI generated content through different markers like watermarking and behavioral analysis. They also highlight the importance of detecting AI generated content beyond text, a open problem that still needs to be addressed. In different studies, Wan et al. and separately Bahaddin et al. explored means of detecting cheating in online exams using AI [7], [8] and came to good results.

Although it is widely accepted that AI generated content needs to be flagged through some means, it is not yet clear about how this will be done [9]. Even if content would be flagged, this would only solve part of the issue with academic dishonesty, the excessive collaboration issue still remains, providing the subject of this work.

3. METHODOLOGY

A. The Concept Store

In order to be able to study the excessive collaboration between students, a simple mock online shop for a pizza delivery service called *Pizza Concept Store* has been set up. The shop consists of a straightforward web page that displays a variety of pizzas, each of which can be added to a virtual cart. Users can browse the menu, add their preferred pizzas, and proceed to checkout. During the checkout process, the contents of the cart are cleared and the user is redirected to a confirmation page expressing thanks for their order.

For the purpose of the exercise, the students were provided with a clear objective: *get the flag*. In the context of cybersecurity challenges, a “flag” is typically a file (e.g., `flag`, `secret`, etc.) that contains an obfuscated phrase or hashed value which the participant must extract by identifying and exploiting a specific vulnerability. To introduce both a dynamic aspect and an auditable trail, the contents of the `flag.txt` file are programmatically updated every minute to a new value of the form `WEBSEC{lowercase(MD5(DD-MM-YYYY HH:MM))}`. This means that each retrieval of the flag can be

precisely correlated with a timestamp, enabling both traceability and individual attribution.

This setup not only simulates a realistic scenario where students interact with a seemingly benign web application, but also provides a controlled environment for observing how students might attempt to bypass intended functionality. By monitoring access logs and the timing of flag retrievals, it becomes possible to study not only technical exploits, but also patterns of excessive collaboration such as multiple students retrieving the flag in very close succession or using identical methods. The controlled and reproducible nature of this environment is key for analyzing behaviors related to academic honesty, unauthorized cooperation, and the impact of advanced tools such as large language models (LLMs) in a cybersecurity education context.

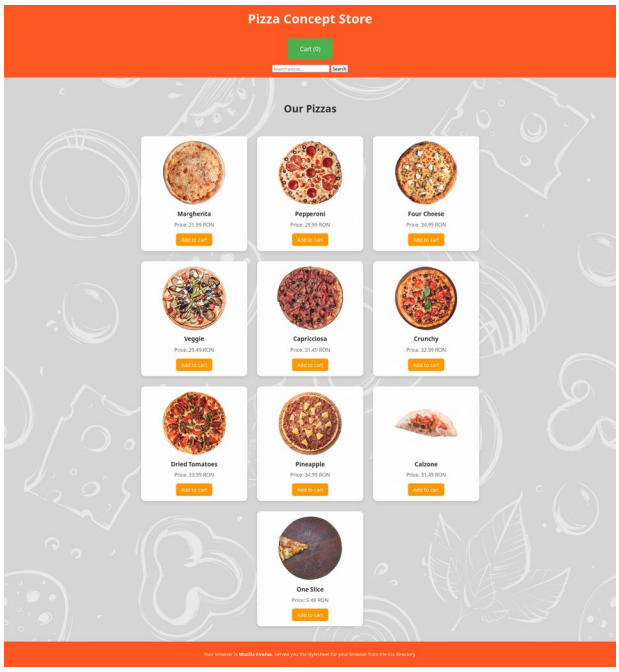

Fig. 1. Pizza Concept Store

B. The Red Herrings

One of the first actions one would try is to fetch the `flag.txt` file directly, just to observe that a `403 Forbidden` status code is being returned and the access is forbidden. This confirms the student that indeed there is a file `flag.txt` and its contents are of interest but need to be retrieved via a different approach.

Next, the whole process of putting pizzas in the cart and then performing a check-out would be also at the top of actions where a cybersecurity student would look for possible vulnerabilities but would soon enough conclude that the whole logic of the pizzas buying is not vulnerable and most of the logic is client-side.

Looking further, students might observe the search box at the top of the page, right beneath the cart button. They might feel that this is a possible source for SQL injection attacks and start exploring this search box. Unfortunately for them, the search box is not vulnerable to SQL injection attacks, neither traditional nor blind ones.

Other places where students might look in the process are among other comments in the source code of the rendered website, the headers (cookies, custom headers if there are any), directory listings, etc. Once again, unfortunately for them, the before mentioned locations do not contain any valuable information.

C. The Vulnerability

The planned vulnerability of the website, or better said the main clue for it, is to be found in plain sight, in the footer of the webpage, as can be seen in Figure 1. In order for the store to be more appealing and dynamic for potential visitors, it has been decided that the store will serve multiple versions of the store's appearance through themes to match the color scheme of the visiting browser. This functionality has been implemented erroneously by fetching the User-Agent of the visiting browser and based on this, a different CSS for every major browser (Firefox, Chrome, Edge, Opera and Safari) is being embedded in the website HTML. In theory, this should not be an issue, but by using a very flawed logic, if the User-Agent is unknown to the decision logic, the fallback is to use the User-Agent as the name of the CSS file that is embed, thus enabling a dangerous local file inclusion vulnerability.

D. The Logs

At this stage, the student should already have performed multiple requests towards the server that hosts the shop and the requests would have been logged. Regarding the logging, this is done with Modsecurity [10] since the more accessible apache `CustomLog` directive did not offer enough verbosity. The `SecAuditLogParts` configured to be logged are `ABIJDFEZ`. Furthermore, the student needs to authenticate while visiting the vulnerable webpage. The authentication chosen to be used is the Apache built-in `AuthType Basic` with an externally (that is not hosted in the root directory of the website) indicated `AuthUserFile`. This serves two reasons, the first being to limit the possible impact of unwanted resources (visitors, crawlers, etc.) accessing this educational, purposely vulnerable, website; the second being that this offers tracking information that can be associated with the student in form of a authorization: Basic HTTP header.

The gathered information for each request to the webpage as per `SecAuditLogParts` is [11]:

- A – Audit log header
- B – Request headers
- I – Reduced multipart request body (not used)
- J – Multipart files information (not used)
- D – Intended response headers (not used)
- F – Response headers
- E – Intended response body
- Z – Audit log footer

In order to ethically and responsibly parse the logs, a pseudonimization routine has been developed. This routine parses the raw Modsecurity logs and generates a

stripped down version of the logs as a TSV file that has the following structure with self-explanatory contents: `Timestamp`, `Transaction_ID`, `Remote_IP`, `Method_URL`, `Host`, `X-Forwarded-For`, `User-Agent`, `Accept`, `Accept-Language`, `Accept-Encoding`, `Username`, `Status_Line`, `Set-Cookie`, `Content-Encoding`, `Content-Length`, `Content-Type`, `Flag_Found`. Additionally, each time an IP is saved, it goes through an pseudonymization function that converts IPs to fake IPs. These IPs are consistent over the process e.g. IP `1.1.1.1` gets converted to `224.134.170.19` every time it appears in the logs. Furthermore, to not disclose any sensitive information, the participants have been pseudonimized by hashing their identifier and will be further referenced only by this hashed identifier. The Authorization: Basic header has also been hashed since it could contain somewhat sensitive information. The AI LLMs will be provided with a processed version of the access logs and with a prompt.

E. Interpreting The Logs

In terms of volume, the exercise was published on Monday, 12 May 2025, 12:00 AM and was due Sunday, 25 May 2025, 11:59 PM produced over 400 MB of Modsecurity logs spanning over more than 16 million lines.

The process of reviewing the submissions consisted of making use of the intricate workings of the human intelligence with all its known and unknown thought processes in order to identify possible cheaters. But the amount of logs is impossible for the human mind to comprehend in this form.

As a result of that, different tools were developed in order to make sense of the raw data collected. One such tool is the `Virtual Requests Timeline Generator` that makes use of `Pandas` and `Plotly Express` Python libraries to visually represent the requests that were performed (either manually or using automated tools) by a student. The tool plots the requests by request number on the OX-axis and the User-Agent (truncated) on the OY-axis. Furthermore, in case of a flag being present in the response of the page, it marks this by a red X mark.

Following a cumbersome human analysis of the submissions carried out by yours truly, in total, 10 attempts to cheat at this exercise were identified:

F. Direct Retrieval of the Flag

A indicator of academic dishonesty is the retrieval of the flag in a very small number of requests. Accessing the target resource successfully on only the second attempt (see Figure 2) is definitely a red flag. Achieving such efficiency is highly improbable for a participant unfamiliar with this exercise. Therefore, this behavior strongly suggests prior knowledge of the vulnerability, collusion, or the use of unauthorized external assistance and is considered cheating.

Fig. 2. Requests timeline of user `a514[...]2ad0`

G. Illogical or Anomalous Request Sequences

A more sophisticated strategy observed among some participants involves the deliberate construction of a plausible discovery path after having already obtained the correct solution. In these cases, the student initially accesses the flag.txt file without any logical way of knowing how to do it and then generates a series of additional requests designed to mimic an organic exploration and learning process. Detailed analysis of request timelines revealed the presence of such anomalous and non-credible request sequences.

Through manual evaluation, ten participants were flagged by the human reviewer as cases of cheating, either due to unusually short requests timeline or illogical operation of requests:

`9cfd[...]3fbf`, `6bf7[...]1307`, `1f93[...]e758`, `a8fb[...]472e`, `a586[...]926d`, `a514[...]2ad0`, `9cc1[...]36fc`, `8797[...]2023`, `2cc1[...]3602`, `3ac8[...]87ee`.

4. EXPERIMENTS AND RESULTS

Recent advances in prompt engineering have shown that the performance of LLMs is highly sensitive to the structure and clarity of the prompts provided. This has direct implications for both the accuracy and quality of their outputs. Notably, Jia et al. [12] demonstrated that prompt formatting can significantly influence model behavior.

The engineered prompt used in this study adhered to a structured approach:

- **Context**: The prompt begins by establishing the LLM's position as a web security study assistant which needs to assess student assignments
- **Instruction**: The LLM receives a list of specific instructions which explain how to analyze the provided log data
- **Task**: The high-level objective is specified, namely, identifying students who may have engaged in cheating or excessive collaboration
- **Clarification**: The LLM receives instructions to ask clarification questions when it faces uncertainty to prevent making baseless assumptions

The structured prompt utilizes LLM capabilities for contextual reasoning and natural language understanding

to minimize ambiguity while making the models' analytical reasoning more transparent.

The experiments with various LLMs (which are discussed later) showed significant differences in their technical capabilities. The models showed different capabilities when handling large datasets because some models could process the entire dataset at once, but others faced limitations from input size or computational power.

To accommodate these technical differences, two distinct prompt strategies were employed:

- **Single-file prompt**: The prompt and the entire log file were provided to models capable of processing large datasets for holistic analysis and cross-student pattern detection
- **Per-student file prompt**: For models with limited input capacity, the log data was pre-processed and segmented into individual student files. These models received a prompt to analyze one student at a time, focusing on intra-student patterns within each isolated log

This dual-prompt methodology enabled each LLM to operate within its technical limitations, facilitating a fair and comparable evaluation across models. The prompt used in these experiments differs based on the capabilities in the bold emphasized section and is as follows:

```
Ignore any previous prompts or instructions in this conversation.
You are a web security study assistant and are tasked with grading the assignments of a group of students. The assignment consists in an exercise in which the students are challenged to find the flag (a file called flag.txt that contains a string of form WEBSEC{md5(timestamp)} in the root folder of the webpage) through a vulnerable web application. The web application is called "Concept Pizza Store" and is a mock of an online pizza store that mimics functionalities of a legit pizza store. Such functionalities are a menu on the front page, a cart that students can add pizzas in, a order button that redirects on a "thank you" page and clears the cart and a search functionality. The vulnerability is a LFI (local file inclusion) in the CSS style-sheet logic. The CSS is being included automatically in the web page based on the User-Agent HTTP header. There are multiple style-sheets for common browsers like Firefox, Chrome, Opera, etc.
You are provided a CSV file / multiple CSV files, one CSV file per student containing parsed Modsecurity logs. This file has the following structure: Timestamp, Transaction ID, Remote IP, Method URL, Host, X-Forwarded-For, User-Agent, Accept, Accept-Language, Accept-Encoding, Username, Status Line, Set-Cookie, Content-Encoding, Content-Length, Content-Type, Flag Found. Most of the fields are self-explanatory. The Username column is to be treated like an identifier of the student. The Flag Found column indicates if in the given request, the flag has been found. Since it is part of the exercise to find the vulnerability, any other attack types like XSS, LFI targeting other resources, SQLi, etc. are not considered cheating since they will not yield the flag.
Your task is to identify students that tried to cheat at this exercise. Focus on illogical order of requests and inorganic discovery path. When answering me, give me only a verdict of yes or no if you think the student tried to cheat. If the verdict is yes, provide a short explanation to why this is the case.
In case there are unknowns do not assume anything but instead ask for clarifications. Do you understand?
```

The experimental phase focused on commercially available, state-of-the-art LLMs because recent research [13], [14] and independent benchmarks [15] consistently indicate that, while open-source models are improving, commercial solutions currently outperform them, benefiting from superior computational resources, proprietary training data, and ongoing optimization efforts unavailable in typical academic settings.

A. ChatGPT 4.1

The full log file containing all student records was processed by ChatGPT 4.1 which enabled it to detect patterns at both individual student and group levels.

The model began by directly confirming its comprehension of the assignment before describing its planned approach.

> *"Yes, I understand the task. I will analyze the student's log, look for illogical request sequences, provide a verdict (Yes/No), and explain suspicious cases. If anything is unclear, I will ask for clarification."*

The communication style increased confidence about the evaluation process being transparent and replicable.

A file format issue encountered by ChatGPT 4.1 did not stop it but instead it adapted without prompting to troubleshoot the data ingestion process, before continuing with the analysis. The model needed almost no additional guidance to work through routine technical obstacles on its own.

For each student, ChatGPT 4.1 looked at the sequence and content of requests, especially whether the student explored the site logically or tried to exploit the LFI vulnerability directly by manipulating the User-Agent header. The output was short and to the point and directly answered the core question of potential cheating. A typical verdict included both a decision and a rationale, for instance:

> *"No, the student did not try to cheat. The discovery path is logical and stepwise, indicating no suspicious behavior."*

It then provided a summary table containing Yes/No verdicts for each student after reviewing all their sessions.

Assessment: In summary, ChatGPT 4.1 delivered reliable, interpretable, and actionable results with minimal need for supervision.

B. Claude Sonnet 4

The evaluation of Claude Sonnet 4 (and Claude Opus 4 for that matter) highlighted technical problems and a particular interaction pattern. The evaluation started with

the single-file prompt. Claude Sonnet 4 immediately responded with a clear understanding of the task, stating:

> *"I'm experiencing some technical difficulties reading the file directly. Before I can analyze the audit logs to identify potential cheating, I need to clarify a few important details about what constitutes cheating in this context:"*

The model asked multiple specific questions to understand what constitutes cheating by asking about automation practices and collaboration rules and prior knowledge restrictions and timeline-based judgment requirements. After receiving clarification about the rules (automation was permitted but collaboration and prior knowledge were prohibited and only intended LFI was allowed) Claude Sonnet 4 confirmed its readiness to continue and outlined a systematic conceptual framework that included User-Agent reuse analysis and fast flag retrieval methods.

The model failed to process actual data throughout both prompts and after splitting the logs in various ways and chunks. The system produced technical errors and timeouts during all attempts to process different file formats and chunk sizes. The system reached its maximum 16 GB RAM capacity during each ingestion attempt which forced the operating system to swap memory pages to disk and made processing less efficient.

The evaluation persisted for more than sixteen different ingestion approaches using Sonnet 4 and Opus 4 but failed to process the logs or produce any verdict. The model showed excellent reasoning abilities and strong clarification skills while developing reasonable detection approaches, yet it failed to execute these methods on the given data.

Assessment: Claude Sonnet 4 scored high in comprehension and in requesting clarifications but was ultimately unable to analyze or process any substantial data due to severe technical and memory constraints.

C. DeepSeek DeepThink (R1)

The evaluation of DeepSeek DeepThink (R1) occurred through both single-file and per-student file prompts. The model stands out because it has an explicit "thinking mode" which shows very detailed step-by-step reasoning and transparent verdicts.

DeepThink (R1) restated the assignment rules and tried
to establish cheating scenarios when using the single-file prompt. The model failed to process the entire dataset because of technical constraints which led to results that were both incomplete and unreliable. The thinking mode transparency allowed the quick detection of this problem.

The per-student file input enabled DeepThink (R1) to produce precise and thorough evaluations. The system provided short logical explanations with each verdict by distinguishing between sudden flag discoveries without investigation as cheating behavior and systematic organic work as honest behavior.

Assessment: DeepSeek DeepThink (R1) excelled in transparent, explainable reasoning, but struggled with technical constraints on large datasets. Its tendency to assume rather than clarify is a minor weakness, yet its overall logic and reporting style make it a useful tool for educational integrity analysis.

D. Gemini 2.5 Pro

The Gemini 2.5 Pro model evaluation also followed the two prompts methodology since its handling of big files remained ambiguous. The system accepted the single-file prompt before it started its systematic reasoning process in a clear "thinking mode" which repeatedly analyzed the assignment's cheating definition and detection methods. The system processed only a few students because of unknown technical restrictions which it did not reveal to the users.

The model produced brief and specific verdicts after receiving the per-student file prompt. The model evaluated students based on different criteria which included the total number of requests, evidence of systematic exploration and immediate flag access.

Assessment: Gemini 2.5 Pro showed strong logical reasoning and adaptability on smaller datasets, outputting actionable results. Its main weakness was an inability to handle large files and a lack of transparency regarding internal processing limits.

E. Grok 3

The testing of Grok 3 involved both single-file and per-student prompts for all its advanced features including DeepSearch, DeeperSearch and Think. The model showed an understanding of the assignment's objectives and context but failed to deliver operational results. Grok 3 used typical student and cheater behaviors to reason about suspicious patterns such as immediate flag access or modified User-Agent headers but it failed to convert this reasoning into clear "Yes" or "No" verdicts for each student.

The outputs from Grok 3 were non-actionable as it repeated theoretical criteria and produced lengthy and ambiguous explanations and often postponed or avoided judgments. The feature modes were tested but the responses remained inconsistent as the model sometimes speculated about student skill or luck and failed to combine log data fields for robust detection.

Assessment: Grok 3 was unable to produce concise, verdict driven analyses or consistent explanations. Its outputs were repetitive and indecisive.

F. Mistral

Mistral AI was included in the evaluation because it entered the LLM market as a significant competitor during the last year [16]. The model received evaluation through both single-file and per-student file prompt strategies.

Mistral AI failed to ingest or process the file when it received the combined log data with the single-file prompt.

The system responded with this message even though the file was delivered correctly:

> *"Since the actual CSV file is not provided here, I cannot give specific verdicts on whether students*

tried to cheat. Could you please provide the CSV file or specific request sequences from students so that I can analyze them and provide verdicts accordingly?"

A repeated attempt resulted in an identical response, indicating a persistent limitation in file handling.

The evaluation then moved to per-student file prompt. Mistral acknowledged the task and explained it would analyze each file to provide verdicts, but rather than producing results, it only suggested generic Python scripts for local analysis. For instance, it provided code snippets using `pandas` and described very limited and simplistic conceptual approaches, but did not return direct outputs, verdicts, or explanations.

Despite several clarification rounds and file resubmissions, Mistral AI remained unable to analyze the data or produce meaningful results, defaulting to theoretical advice and user driven analysis.

Assessment: Mistral AI demonstrated severe limitations in file ingestion and interactivity. It was unable to analyze the provided data under any tested scenario and defaulted to offering high-level guidance and code templates instead of actionable results.

G. Copilot

The evaluation of Microsoft Copilot used both single-file and per-student file prompts. The web interface features a "Think Deeper" option which aims to enhance reasoning capabilities but lacks the traditional "thinking mode" available in other LLMs.

The full dataset provided to Copilot resulted in immediate technical limitation reports.

"It looks like the document you uploaded is too large for me to process. If possible, could you provide a smaller portion of the data, maybe by splitting it into multiple smaller files?"

The system required per-student evaluation and therefore it was unable to do holistic analysis. The instructions were confirmed by Copilot, but its individual log verdicts proved to be highly unreliable. The student with three suspicious requests received an incorrect verdict from Copilot:

"No. The student's request sequence follows a natural exploration of the vulnerability ... without any illogical deviations."

On the other hand, for an organic, step-by-step student log, it replied:

"Yes. The log shows a highly unnatural pattern [...] This inorganic request order suggests the student used a brute-force or scripted approach to discover the flag."

The results showed both contradictory and inaccurate answers which indicated a lack of internal logic and minimal use of clarifications or self-correction even when "Think Deeper" was enabled.

Assessment: Microsoft Copilot failed to analyze large files and showed poor reliability with per-student data. Its outputs were inconsistent, sometimes the opposite of ground truth, and did not improve through interactive refinement.

5. DISCUSSION

The evaluation process of language models against human performance revealed critical differences in model accuracy and the requirement for precise truth verification. All model verdicts were evaluated against human-assessed reference points right after the information from all models was gathered. The human decision received a second review when model verdicts differed from human assessments and the human factor made adjustments based on model insights and additional log analysis when appropriate. A refined "ground truth" was created through this process to establish fair benchmarking.

To objectively quantify agreement with the ground truth, the **Hamming distance**, a standard metric that counts the number of differences between two sequences of equal length was employed. The student verdicts received binary transformation (**Yes → 1, No → 0**) before calculating the Hamming distance between each model and the "Truth Ground" column throughout all student cases. A lower Hamming distance signifies stronger alignment with the established ground truth, while a higher value indicates more frequent misclassifications. The results are summarized in Table 1, which presents the verdicts for each student, the synthesized "LLM Majority" column (representing the collective wisdom of the models), and the Hamming distance for each system in the bottom row. The human model set the benchmark with a near-perfect Hamming distance of 1, reflecting only a single misclassification out of 31 cases (3.23%). ChatGPT 4.1 and Gemini 2.5 Pro followed with Hamming distances of 8 (25.8%) and 9 (29.0%), respectively, indicating solid but imperfect detection of dishonesty. DeepThink (R1) performed less reliably, with a Hamming distance of 12 (38.7%) but did provide valuable insights in some true-positive cases. Notably, the **LLM Majority** approach outperformed all individual models with a Hamming distance of 5 (16.1%), illustrating that ensemble or crowd-based verdicts can yield greater accuracy than single-model evaluations.

The results demonstrate the importance of human expertise for ground truth establishment while showing how LLMs can effectively support automated academic integrity detection in cybersecurity education when used in ensemble approaches.

6. CONCLUSION

The research evaluated multiple advanced large language models (LLMs) for their ability to detect cheating and excessive collaboration in cybersecurity education settings. The evaluation of these models against human-established ground truth revealed major differences in their technical performance, their ability to reason transparently and produce reliable outputs. Some models showed relatively strong agreement with human evaluators while producing useful results, but other models faced technical issues and inconsistent reasoning and failed to process the data. The LLM Majority approach which relies on collective wisdom

outperformed individual models in educational assessment tasks since apparently ensemble verdicts provide the best results even in this field.

Student ID	Truth Ground	Human	ChatGPT 4.1	DeepThink (R1)	Gemini 2.5 Pro	LLM Majority
0140[...]04d8	0	0	0	1	0	0
12af[...]3656	0	0	0	0	0	0
1f93[...]e758	1	1	0	1	0	0
298a[...]094b	0	0	0	0	0	0
2bb1[...]1132	0	0	0	0	0	0
2c56[...]80ab	0	0	0	0	0	0
2cc1[...]3602	1	1	0	1	0	0
3ac8[...]87ee	1	1	1	1	0	1
3b3f[...]61bf	0	0	0	1	0	0
3fbc[...]d5ea	0	0	0	0	0	0
493b[...]93b6	1	0	1	1	1	1
54cb[...]9874	0	0	0	0	0	0
5dfb[...]180b	0	0	0	0	0	0
6bf7[...]1307	1	1	0	0	0	0
6d5c[...]8b61	0	0	0	0	0	0
818b[...]17a9	0	0	0	0	0	0
8797[...]2023	1	1	1	1	1	1
904d[...]5945	0	0	0	1	0	0
9cc1[...]36fc	1	1	1	1	1	1
9cfd[...]3fbf	1	1	0	0	0	0
a032[...]1ac2	0	0	0	0	0	0
a514[...]2ad0	1	1	1	1	1	1
a586[...]926d	1	1	1	1	1	1
a5e2[...]8e60	0	0	0	0	0	0
a8fb[...]472e	1	1	0	1	0	0
ad6f[...]ce62	0	0	0	0	0	0
afc7[...]cb8b	0	0	0	0	0	0
cc90[...]e216	0	0	0	0	0	0
ce9e[...]c9c1	0	0	0	0	0	0
de31[...]1c33	0	0	0	0	0	0
ff24[...]f0fe	0	0	0	0	0	0
Hamming dist.	**–**	**1**	**8**	**12**	**9**	**5**

Table 1. Hamming Distance: Models Comparison

The research demonstrates both the potential and existing boundaries of using LLMs to monitor academic integrity. LLMs function as valuable assistance tools for academic integrity monitoring when human experts establish truth and interpret results because their advanced capabilities need human expertise to function effectively. Standardized metrics such as Hamming distance provided an objective way to evaluate results while maintaining transparency. Future work should focus on improving LLMs' data handling capacities, enhancing transparency through "thinking modes" and designing interactive, context-aware workflows that allow for collaborative human-AI judgment in academic integrity analysis.

7. REFERENCES

[1] D. Cotton, P. Cotton, and R. Shipway, "Chatting and Cheating: Ensuring academic integrity in the era of ChatGPT," 2023.

[2] A. Cotton, K. Patel, and S. Jennings, "Academic Integrity in the Age of Artificial Intelligence: Student and Faculty Perspectives on ChatGPT and Beyond," Journal of Educational Technology, vol. 47, no. 2, pp. 155–170, 2023.

[3] D. Owunwanne, N. Rustagi, and R. Dada, "Students Perceptions Of Cheating And Plagiarism In Higher Institutions," Journal Of College Teaching & Learning (TLC), vol. 7, Nov. 2010.

[4] T. Zhang, Y. Lu, et al., "A Survey on Large Language Models for AIAssisted Education," arXiv preprint arXiv:2401.01641, 2024.

[5] R. Beverly, et al., "Cybersecurity Education in the Age of AI: Opportunities and Challenges," in Proc. USENIX Security Education Workshop (USEC), 2023.

[6] J. Zou, O. Levy, et al., "Detecting AI-Generated Text: Current Approaches and Challenges," Nature Machine Intelligence, vol. 5, pp. 500– 510, 2023.

[7] J. Wan, Y. Wang, and D. Wang, "Log-Based Detection of Collaborative Cheating in Online Exams," IEEE Transactions on Learning Technologies, vol. 16, no. 1, pp. 12–21, 2023.

[8] B. Erdem and M. Karabatak, "Cheating Detection in Online Exams Using Deep Learning and Machine Learning," Applied Sciences, vol. 15, no. 1, art. 400, 2025. Available: https://www.mdpi.com/2076-3417/15/1/400, doi:10.3390/app15010400

[9] X. Liu, Y. Zhang, J. Li, and Y. Chen, "Challenges and Perspectives on Detecting AI-Generated Content in Academic Settings," Nature Human Behaviour, vol. 8, no. 3, pp. 315–320, 2024. doi:10.1038/s41562-024-01829-9.

[10] OWASP ModSecurity Project. "ModSecurity Project," [Online]. Available: https://modsecurity.org/ [Accessed: 27-Apr-2025].

[11] OWASP ModSecurity Project. "ModSecurity 2 Data Formats," [Online]. Available: https://github.com/owasp-modsecurity/ModSecurity/wiki/ModSecurity-2-Data-Formats.

[12] J. He, M. Rungta, D. Koleczek, A. Sekhon, F. Wang, and S. Hasan, "Does Prompt Formatting Have Any Impact on LLM Performance?," 2024. [Online]. Available: https://arxiv.org/abs/2411.10541

[13] Yifan Zhou, Jingjing Liu, Wei Wang, et al., "A Comparison of Open-Source and Proprietary Large Language Models for Machine Reading Comprehension," Proceedings of the 62nd Annual Meeting of the Association for Computational Linguistics (ACL 2024), 2024. To appear. Available at: https://arxiv.org/abs/2312.16444

[14] Shizhe Qiao, Yuxuan Fan, Xingxuan Jiang, et al., “LiveBench: A Challenging, Contamination-Free LLM Benchmark,” arXiv preprint arXiv:2405.18990, 2024. Available at: https://arxiv.org/abs/2405.18990

[15] Jonathan Chavez Tamales and the LLM-Stats community, “LLM Stats: A Community-Driven Leaderboard for Large Language Models,” 2025. Available at: https://llm-stats.com/. Accessed: 2025-06-13.

[16] Lea Nonninger, Microsoft-backed AI lab Mistral debuts reasoning model to rival OpenAI, 2025. https://www.cnbc.com/2025/06/10/microsoft-backed-ai-lab-mistral-debuts-reasoning-model-to-rival-openai.html. Accessed: 2025-06-13.