



International Journal of Recent Development in Engineering and Technology
Website: www.ijrdet.com (ISSN 2347-6435 (Online) Volume 15, Issue 10, October 2026)

Integrity AI: An Automated Academic Plagiarism and AI-Content Detection

Dr. Baby Kalpana Y¹, Mohan Kumar M², Pranav Karthik S³, Raghul R⁴, Nidhish Kumar J⁵, Mohamed Athil M⁶

¹Professor, Department of Computer Science and Engineering, Sri Shakthi Institute of Engineering and Technology, Coimbatore – 641062, India

^{2,3,4,5,6}Department of Computer Science and Engineering, Sri Shakthi Institute of Engineering and Technology, Coimbatore – 641062, India

Abstract— The wide availability of large language models has made it harder for educators to verify the originality of student submissions, because a text may be copied from an existing source or generated by an AI writing assistant. This paper presents IntegrityAI, a web-based platform that screens academic documents for both source-based similarity and AI-generated writing patterns within a single submission workflow. Students upload TXT, PDF or DOCX files through a role-based web interface; a Python Flask back end extracts the text, segments it into paragraphs and sentences, and encodes each sentence with the all-MiniLM-L6-v2 sentence-transformer model. Every sentence is compared by cosine similarity with an embedded reference corpus, and sentences whose best match reaches 0.75 are flagged together with the matched source citation. In parallel, each paragraph receives a heuristic AI-content score based on sentence-length variation, vocabulary diversity and average word length. A report generator combines both signals into a four-level risk classification with confidence factors, sentence-level heatmap data and recommendations for faculty, and all users, documents and scan results are stored in MongoDB. The paper describes the architecture, algorithms and data model of the implemented prototype, discusses its current limitations, including a small seed corpus and a heuristic rather than trained AI detector, and outlines the evaluation required before institutional use.

Keywords—Academic integrity, AI-generated text detection, MongoDB, plagiarism detection, semantic similarity, sentence transformers, stylometry

I. INTRODUCTION

Plagiarism detection has been studied for decades, and the techniques used range from exact-match document fingerprinting to methods that compare the meaning of text [1]. The arrival of large language models has added a second and harder problem: an assignment can now be generated rather than copied, so there may be no single source document to match. Independent testing of publicly available detection tools has shown that they are neither fully accurate nor reliable [2], and some detectors have been reported to misclassify writing by non-native English speakers as machine-generated [3].

Faculty members therefore need support that addresses both concerns and that presents evidence, such as matched sentences and paragraph-level indicators, rather than a single unexplained percentage. IntegrityAI was developed with this aim as a prototype platform in which a student submits a document once and faculty members receive a combined report on source similarity and AI-writing indicators.

The contributions of this paper are as follows:

- a unified submission pipeline that applies sentence-level semantic similarity and paragraph-level AI-content estimation to the same extracted text;
- a role-based web platform for students, faculty members and administrators, including administrator approval of self-registered accounts;
- a report format that combines a risk level, confidence factors, heatmap data and recommendations; and
- a candid account of the prototype's limitations together with an evaluation plan.

Section II reviews related work, Section III states the problem and objectives, Sections IV to VI describe the architecture, implementation and data model, Section VII describes the experimental setup, Section VIII discusses the results, Section IX lists advantages, limitations and applications, and Section X concludes the paper.

II. RELATED WORK

A. Plagiarism Detection

Foltýnek et al. [1] classify academic plagiarism detection methods into lexical, syntactic, semantic and other approaches. Fingerprinting algorithms such as winnowing [4] select hashes of overlapping character n-grams and detect verbatim copying efficiently, but they are less effective when the copied text has been paraphrased. Semantic approaches instead represent text as dense vectors. Sentence-BERT [5] produces sentence embeddings that can be compared directly with cosine similarity, and knowledge distillation methods such as MiniLM [6] yield compact transformer models that are suitable for running on ordinary hardware.

IntegrityAI uses a compact MiniLM-based model distributed with the Sentence-Transformers library [5].

B. AI-Generated Text Detection

Supervised detectors fine-tune a pre-trained encoder, for example RoBERTa [7], on human and machine-generated text; OpenAI released such a detector for GPT-2 output [8]. Statistical approaches score a text with a language model such as GPT-2 [9]: GLTR visualises how often the words of a text fall among the model’s top-ranked predictions [10], and DetectGPT uses the curvature of the model’s log-probability function as a zero-shot signal [11]. Several studies question the reliability of all of these detectors: paraphrasing can evade them [12], tested tools showed limited accuracy [2], and false positives fall unevenly on different groups of writers [3]. Variation in sentence length, often called “burstiness”, is also used informally as a stylometric cue, but it is a weak signal on its own. IntegrityAI does not propose a new detection algorithm; it integrates established components into a single workflow and presents its outputs as probabilistic indicators for human review.

III. PROBLEM STATEMENT AND OBJECTIVES

A. Problem Statement

Universities face a dual challenge: conventional plagiarism and the submission of AI-generated assignments that are difficult to distinguish from student writing by inspection. The task addressed in this work is to build a system that checks submitted documents for both source-based plagiarism and AI-generated content patterns, and that generates a transparent, evidence-backed report for faculty review.

B. Objectives

- Detect sentences that are semantically similar to known sources and cite the matched source.
- Estimate a heuristic AI-content score for every paragraph of a submission.
- Produce a faculty-facing report with a risk level, confidence factors and recommendations.
- Provide separate, role-based access for students, faculty members and administrators.
- Reduce the manual effort of screening submissions, with a target processing time below 20 seconds per document.

IV. PROPOSED SYSTEM AND ARCHITECTURE

A. Architecture Overview

IntegrityAI follows a three-tier architecture, shown in Fig. 1. The presentation layer is a single-page web application written in HTML, CSS and JavaScript and served by the back end. It provides a student portal, a faculty dashboard and an administrator console, and communicates with the server through JSON requests. The application layer is a Python Flask REST API that handles authentication, document submission, report retrieval, user management, statistics and literature search. The detection layer comprises four modules: text extraction, the plagiarism engine, the AI-content detector and the report generator. Persistent data are stored in MongoDB, and the literature-search endpoint calls the Semantic Scholar and CrossRef web APIs.

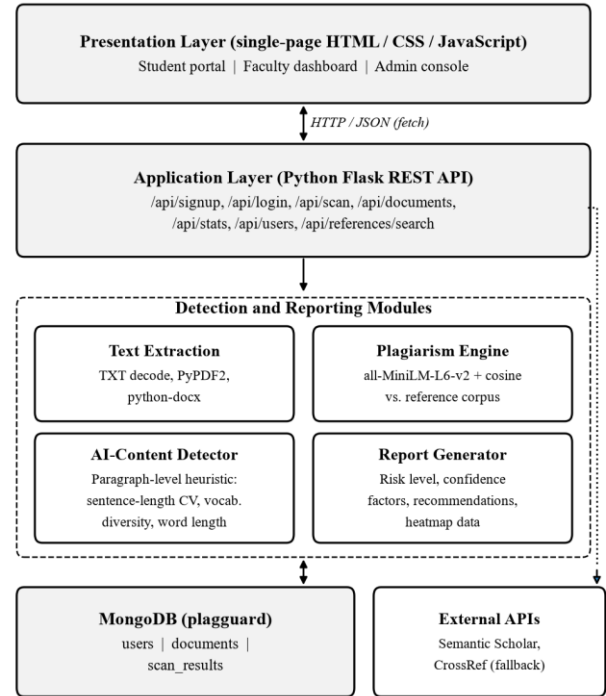


Fig. 1. System architecture of IntegrityAI

Table I lists the technologies used in the implemented prototype, as declared in the project’s dependency file and source code.

TABLE I
IMPLEMENTED TECHNOLOGY STACK

Layer	Technology
Front end	HTML, CSS, JavaScript (single page), Font Awesome icons
Back end	Python, Flask 3.1, Flask-CORS
Text extraction	PyPDF2 3.0.1, python-docx 1.1.2
Similarity model	sentence-transformers 4.1.0 (all-MiniLM-L6-v2), PyTorch 2.6
AI-content scoring	Rule-based heuristics implemented with NumPy
Database	MongoDB accessed through PyMongo
Password hashing	bcrypt
Literature search	Semantic Scholar Graph API, CrossRef REST API

A. User Roles and Access Control

The platform recognises three roles: student, faculty and admin. During registration a user supplies a full name, an e-mail address, a password, an optional phone number and a role. Passwords are hashed with bcrypt before storage, with SHA-256 used only if the bcrypt library is unavailable. Self-registered accounts are created with the status “pending”, and an administrator must set the status to “active” before the user can sign in; accounts can also be rejected or suspended. At sign-in, the server verifies the password, checks that the role selected on the login page matches the stored role, refuses pending and rejected accounts, and records the user identifier and role in the Flask session. Table II summarises the main REST endpoints.

TABLE II
MAIN REST API ENDPOINTS

Endpoint	Method	Purpose
/api/signup	POST	Register an account (pending approval)
/api/login	POST	Role-checked sign-in
/api/scan	POST	Upload text or a file and run both analyses
/api/documents	GET	List documents with latest scores
/api/documents/<id>/report	GET	Retrieve the latest full report
/api/users/pending	GET	List accounts awaiting approval
/api/users/<id>/status	PATCH	Approve, reject or suspend a user
/api/stats	GET	Aggregate system statistics
/api/references/search	POST	Find related academic papers
/api/health	GET	Service and model status

V. METHODOLOGY AND IMPLEMENTATION

Fig. 2 shows the processing workflow executed for every submission.

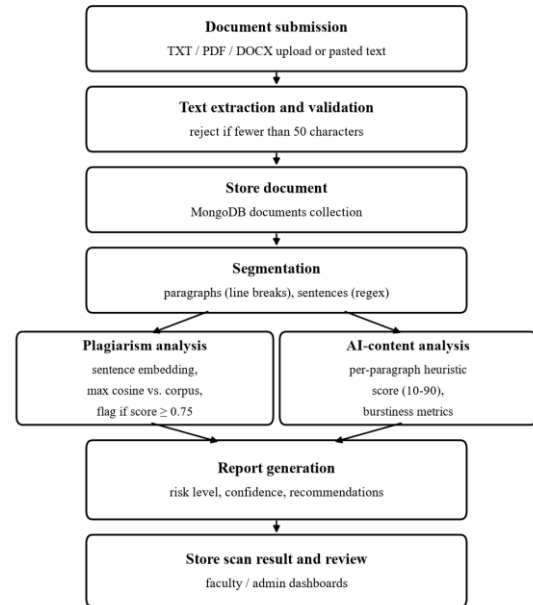


Fig. 2. Document scanning workflow

A. Document Ingestion

The scan endpoint accepts either an uploaded file or pasted text. Plain-text files are decoded as UTF-8, PDF files are read page by page with PyPDF2, and DOCX files are read paragraph by paragraph with python-docx. Inputs with fewer than 50 characters are rejected, and an error is returned when no text can be extracted from a PDF. The word count is computed, and the document is stored in MongoDB before analysis begins.

B. Text Segmentation

Paragraphs are obtained by splitting the text on line breaks. The plagiarism engine keeps paragraphs longer than 20 characters and the AI-content detector keeps paragraphs longer than 30 characters. Sentences are obtained with a regular expression that splits after a full stop, question mark or exclamation mark followed by white space, and sentences of 10 characters or fewer are discarded.

C. Plagiarism Detection Engine

The engine loads the all-MiniLM-L6-v2 model on first use.

It then builds a one-time index by splitting each reference passage into sentences, encoding them and storing, for each sentence, the title, author, source and URL of its passage. For an input sentence s with embedding e_s and a corpus sentence c with embedding e_c , the similarity is the cosine of the two vectors:

$$\text{sim}(s, c) = (e_s \cdot e_c) / (\|e_s\| \|e_c\|) \quad (1)$$

The best score of a sentence is the maximum of (1) over all corpus sentences, and the sentence is flagged when this score is at least $\tau = 0.75$, in which case the matched sentence and its source metadata are attached as a citation. A paragraph score is the mean best score of its sentences, and the document-level similarity score is the mean of the best sentence-level similarity scores, expressed on a percentage scale. This score represents semantic similarity to the available reference corpus and should not be interpreted as a direct percentage of plagiarised content. The report also gives the percentage of flagged sentences and, for every matched source, the number of matches and the average similarity. Each sentence is assigned a heatmap colour band: 0.85 and above, 0.75–0.85, 0.60–0.75, 0.40–0.60 and below 0.40. The current reference corpus is a seed set of 15 passages on computing topics, including data structures, machine learning, databases, operating systems, networks, software engineering, security, cloud computing, natural language processing, academic plagiarism, research methodology and computing ethics.

D. AI-Content Detection

The detector assigns each paragraph a heuristic AI-content score based on three stylometric features. The score starts at 30. An additional 20 points are assigned when the coefficient of variation of sentence lengths is below 0.25; otherwise, 10 points are assigned when it is below 0.40. A further 10 points are assigned when the type–token ratio lies between 0.45 and 0.65, and 5 points are assigned when the mean word length exceeds 5.5 characters. Under these rules, the paragraph score ranges from 30 to 65. Scores of 55 or above indicate that the heuristic threshold for possible AI-writing patterns has been reached; scores below 55 indicate that it has not been reached. The document-level score is calculated as the mean of its paragraph scores. The coefficient of variation of sentence lengths, multiplied by 100, is reported as a burstiness measure. A vocabulary-based perplexity estimate is reported as a supporting metric but does not contribute to the heuristic score. Neither supporting metric contributes to the heuristic score.

An alternative pathway for a RoBERTa classifier and GPT-2 perplexity using the Hugging Face Transformers library [14] is present in the source code but disabled in the current version. Therefore, the current detector uses heuristic scoring rather than a trained and validated classification model. Its output is a heuristic indicator, not a calibrated probability of AI-generated content.

E. Report Generation and Risk Classification

Let P be the plagiarism score and A the AI-content score. The report generator computes a combined risk score of $0.5P + 0.5A$ and classifies the overall risk using the larger score $m = \max(P, A)$ and the mean $a = (P + A)/2$, as listed in Table III. Three confidence factors are reported: sample size (high for at least 20 sentences, medium for at least 10), the AI model in use (low in heuristic mode) and the consistency of the two signals (high when they differ by less than 15 points, medium below 30). Rule-based recommendations are added according to the scores, and every report includes a disclaimer that AI scores are probabilistic estimates and not proof of misconduct. The report also records whether the processing time stayed below the 20-second target.

TABLE III
DECISION THRESHOLDS USED IN THE PROTOTYPE

Decision	Rule
Sentence flagged as similar	cosine similarity ≥ 0.75
Possible AI-writing patterns	AI-content score ≥ 55
Heuristic threshold for possible AI-writing patterns not reached	AI-content score < 55
Critical risk	$m \geq 80$ or $a \geq 70$
High risk	$m \geq 60$ or $a \geq 50$
Medium risk	$m \geq 40$ or $a \geq 30$
Low risk	otherwise

F. Related-Literature Search

Faculty members and administrators can request a list of related papers for a document. The server extracts up to eight frequent words of four or more letters, excluding stop words, optionally prefixes the subject area, and queries the Semantic Scholar paper-search API [13] for six results, falling back to the CrossRef API if that request fails. Titles, authors, years, links or DOIs and abstract excerpts are returned. These results help the reviewer but are not used in the similarity score.

G. User Interface

Students upload TXT, PDF or DOCX files or paste text, select a subject and view their submission history; after a scan they see a confirmation rather than the scores.

Faculty members browse submitted documents and open a report showing the plagiarism and AI-content scores, the sentence heatmap, matched sources and related papers. Administrators approve or reject registrations, manage users, review scan logs and view aggregate statistics such as the number of scans, average scores and the number of high- or critical-risk documents.

VI. DATABASE DESIGN

MongoDB stores the data in a database named “plaggard” with the three collections listed in Table IV. A unique index on the e-mail field prevents duplicate accounts, and indexes on the user, document and timestamp fields support the dashboard queries. The full report is embedded in each scan result as a sub-document after NumPy values are converted into BSON-compatible types.

TABLE IV
MONGODB COLLECTIONS

Collection	Main fields
users	full_name, email (unique), phone, password_hash, role, status, created_at, updated_at
documents	user_id, filename, original_text, word_count, subject, uploaded_at
scan_results	document_id, plagiarism_score, ai_score, risk_level, report_json, processing_time, created_at

VII. EXPERIMENTAL SETUP AND EVALUATION

A. Software Environment

The prototype runs as a single Flask process on port 5000 and connects to a local MongoDB server by default; the connection string and database name can be overridden with environment variables. Library versions are those listed in Table I.

The IntegrityAI prototype was developed and tested on two separate laptop systems to verify its functionality across different computing environments.

System 1 – Developer's Laptop: The first system was an ASUS TUF Gaming A15 laptop equipped with an AMD Ryzen 7 7435HS processor, 16 GB RAM, and an NVIDIA GeForce RTX 3050 GPU with 4 GB VRAM.

System 2 – Team Member's Laptop: The second system was an Acer Aspire Lite laptop equipped with an AMD Ryzen 5 series processor, 16 GB RAM, and 512 GB storage.

The prototype uses Python with the Flask web framework and MongoDB as the database. The application was executed locally on both systems for functional verification.

B. Functional Verification

The repository contains a script that submits a short computer-science paragraph to the scan endpoint and prints the resulting plagiarism score, AI-content score and risk level. A separate script checks the integrity of the front-end page.

Functional testing was conducted on two laptop systems to verify the operation of the implemented features. The tests covered user registration and authentication, document upload, text extraction, plagiarism similarity analysis, paragraph-level AI-content scoring, report generation, and role-based access control. The integrated submission-to-report workflow was also checked. These tests demonstrate the functional operation of the prototype; however, they do not establish quantitative detection accuracy or processing-time performance.

C. Illustrative Output of the AI-Content Module

To illustrate the behaviour of the heuristic, the AI-content module was run on its own on the sample paragraph contained in the test script (two sentences, 45 words). Both sentences have similar lengths (mean 22.5 words, standard deviation 1.5), giving a burstiness value of 6.7, so the uniformity rule added 20 points; the long-word rule added 5 more, and the paragraph received a score of 55.0 and the verdict “possibly AI-generated”. This is a single unlabelled example that shows how the rules operate; it is not a measure of accuracy, and it also shows that very short inputs can easily appear uniform.

D. Evaluation Plan

Although functional testing was conducted on two laptop systems, quantitative evaluation remains necessary to determine the detection performance of the prototype. Future evaluation should use labelled datasets to measure plagiarism detection precision and recall and AI-content detection performance against appropriate baselines. Repeated processing-time measurements should also be collected to determine whether the target of less than 20 seconds per document is consistently achieved.

VIII. RESULTS AND DISCUSSION

The implemented IntegrityAI prototype integrates document submission, text extraction, sentence-level semantic similarity analysis against the available reference corpus, paragraph-level heuristic AI-content scoring, combined risk classification, report generation, and persistent storage of results. The plagiarism module uses sentence embeddings to identify semantic similarities between submitted text and available reference passages.



The resulting scores, confidence factors, and recommendations are presented as indicators for human review.

Functional testing examined the operation of the submission-to-report workflow on two laptop systems. In the illustrative test described in Section VII-C, the AI-content module produced a score of 55.0 and the verdict “possibly AI-generated”. This example demonstrates the operation of the heuristic scoring rules; it does not establish detection accuracy. The plagiarism score reflects similarity to the available reference corpus and should not be interpreted as a direct percentage of plagiarised content.

A formal quantitative evaluation has not yet been completed. Plagiarism detection precision and recall, AI-content detection performance, and processing time across repeated document submissions have not been measured in a controlled evaluation. Evaluation using labelled datasets, suitable comparison baselines, and repeated processing-time measurements is required to assess the system’s performance and determine whether the target processing time of less than 20 seconds per document is consistently achieved.

IX. ADVANTAGES, LIMITATIONS AND APPLICATIONS

A. Advantages

- A single upload produces both a similarity analysis and an AI-content analysis.
- Flagged sentences are shown with the matched source sentence and citation.
- Paragraph-level AI scores show which part of a document needs attention.
- Role-based dashboards and administrator approval support an institutional workflow.
- Reports are stored, so they can be reviewed again later.

B. Limitations

The current prototype has the following limitations:

- The reference corpus contains only 15 seed passages, so similarity can be detected only against those topics; papers found through Semantic Scholar are not added to the index.
- The AI-content detector is a heuristic that has not been trained or validated; it tends to rate short or formally written text as uniform and is subject to the known weaknesses of AI detectors [2], [3], [12].

- If the sentence-transformer model fails to load, the current implementation may return simulated similarity scores. These scores do not represent genuine similarity analysis and must not be interpreted as detection results. This fallback should be removed or disabled before real-world use.
- Quoted and properly cited text is not excluded from the similarity score, only English is supported, and scanned PDFs without a text layer cannot be analysed.
- The current prototype has security limitations. Some administrative endpoints do not enforce server-side role checks, the Flask session secret is fixed in the source code, demonstration accounts with known passwords are created at start-up, and the password-reset interface is not connected to the back end. These issues must be addressed before deployment with real student data.
- No quantitative evaluation has been completed yet.

C. Applications

After the limitations above are addressed and the evaluation is complete, the platform can be used to pre-screen assignments in colleges and universities, to support faculty members during the review of suspicious submissions, to monitor integrity trends at the department level through the administrator statistics, and as a teaching aid that shows students how close their wording is to existing sources.

X. CONCLUSION AND FUTURE WORK

This paper presented IntegrityAI, a role-based web platform that integrates sentence-embedding-based semantic similarity analysis and paragraph-level stylometric heuristics to screen academic submissions for potential plagiarism and AI-generated writing patterns. The implemented prototype supports document submission, text extraction, analysis, report generation and persistent storage of results in MongoDB. Functional testing was conducted on two laptop systems to verify the operation of the main workflow. However, the reference corpus remains limited to 15 seed passages, and the AI-content detector uses heuristic rules rather than a validated classification model. Therefore, the generated scores should be treated as preliminary indicators for human review, not as definitive evidence of academic misconduct. Further evaluation using labelled datasets and repeated processing-time measurements is required before the system can be considered for institutional deployment.



International Journal of Recent Development in Engineering and Technology
Website: www.ijrdet.com (ISSN 2347-6435 (Online) Volume 15, Issue 10, October 2026)

Future work will address these points:

- expand the reference corpus with open-access papers retrieved from Semantic Scholar and index them with a vector search library such as FAISS;
- enable and evaluate a trained classifier and model-based perplexity in place of the heuristic AI-content score;
- exclude quoted and cited text from the similarity calculation;
- enforce role checks on every endpoint and move secrets to configuration;
- export reports as PDF files with the heatmap overlaid on the original document;
- carry out the evaluation described in Section VII; and
- add learning-management-system integration and multilingual support.

Acknowledgment

The authors thank the Department of Computer Science and Engineering, Sri Shakthi Institute of Engineering and Technology, Coimbatore, for the facilities and support provided during this project.

REFERENCES

- [1] T. Foltýnek, N. Meuschke, and B. Gipp, "Academic plagiarism detection: A systematic literature review," *ACM Computing Surveys*, vol. 52, no. 6, Art. 112, 2019.
- [2] D. Weber-Wulff, A. Anohina-Naumeca, S. Bjelobaba, T. Foltýnek, J. Guerrero-Dib, O. Popoola, P. Šigut, and L. Waddington, "Testing of detection tools for AI-generated text," *International Journal for Educational Integrity*, vol. 19, Art. 26, 2023.
- [3] W. Liang, M. Yuksekgonul, Y. Mao, E. Wu, and J. Zou, "GPT detectors are biased against non-native English writers," *Patterns*, vol. 4, no. 7, Art. 100779, 2023.
- [4] S. Schleimer, D. S. Wilkerson, and A. Aiken, "Winnowing: Local algorithms for document fingerprinting," in *Proc. ACM SIGMOD Int. Conf. Management of Data*, 2003, pp. 76–85.
- [5] N. Reimers and I. Gurevych, "Sentence-BERT: Sentence embeddings using Siamese BERT-networks," in *Proc. EMNLP-IJCNLP*, 2019, pp. 3982–3992.
- [6] W. Wang, F. Wei, L. Dong, H. Bao, N. Yang, and M. Zhou, "MiniLM: Deep self-attention distillation for task-agnostic compression of pre-trained transformers," in *Advances in Neural Information Processing Systems*, vol. 33, 2020, pp. 5776–5788.
- [7] Y. Liu et al., "RoBERTa: A robustly optimized BERT pretraining approach," arXiv:1907.11692, 2019.
- [8] I. Solaiman et al., "Release strategies and the social impacts of language models," arXiv:1908.09203, 2019.
- [9] A. Radford, J. Wu, R. Child, D. Luan, D. Amodei, and I. Sutskever, "Language models are unsupervised multitask learners," OpenAI, Technical Report, 2019.
- [10] S. Gehrmann, H. Strobelt, and A. M. Rush, "GLTR: Statistical detection and visualization of generated text," in *Proc. 57th Annual Meeting of the ACL: System Demonstrations*, 2019, pp. 111–116.
- [11] E. Mitchell, Y. Lee, A. Khazatsky, C. D. Manning, and C. Finn, "DetectGPT: Zero-shot machine-generated text detection using probability curvature," in *Proc. 40th Int. Conf. Machine Learning*, PMLR vol. 202, 2023, pp. 24950–24962.
- [12] V. S. Sadasivan, A. Kumar, S. Balasubramanian, W. Wang, and S. Feizi, "Can AI-generated text be reliably detected?" arXiv:2303.11156, 2023.
- [13] R. Kinney et al., "The Semantic Scholar open data platform," arXiv:2301.10140, 2023.
- [14] T. Wolf et al., "Transformers: State-of-the-art natural language processing," in *Proc. EMNLP 2020: System Demonstrations*, 2020, pp. 38–45.