

DeltaDetect: AI-Powered Plagiarism Detection and Automated Report Generation

Mr. Shahu Nikam¹ Mr. Swarup Bonde² Ms. Shreya Bhavsar³ Ms. Shravani Kawade⁴
Prof. R. V. Shinde⁵

^{1,2,3,4}*Department of Information Technology MVP's Rajarshi Shahu Maharaj Polytechnic, Gangapur Road, Nashik, Maharashtra, India*

⁵*Professor, Department of Information Technology MVP's Rajarshi Shahu Maharaj Polytechnic, Gangapur Road, Nashik, Maharashtra, India*

Abstract—Academic plagiarism has become harder to detect because paraphrasing tools and generative AI systems such as ChatGPT and Gemini can produce text that keyword-matching checkers do not recognize, while capable commercial tools are costly and need documents to be uploaded to external servers. This paper presents DeltaDetect, a proposed AI-powered plagiarism detection system that combines TF-IDF keyword vectorization with Sentence-BERT (SBERT) semantic embeddings in a weighted hybrid similarity score, and adds a self-trained offline classifier for AI-generated content that needs no third-party API. The findings are compiled automatically into a PDF report with a similarity percentage, an AI-content score, highlighted passages and matched sources. The paper describes the architecture, modules, working principle and planned evaluation of the system, which is intended to give institutions a private, free-to-use and explainable originality-verification tool.

Index Terms—Plagiarism Detection, TF-IDF, Sentence-BERT, Semantic Similarity, AI-Generated Content Detection, Report Generation

I. INTRODUCTION

Digital content has made it easy to find, copy and reuse other people's writing. Plagiarism, the presentation of someone else's words or ideas as one's own, damages the credibility of institutions and discourages honest students. Two developments have made it harder to detect. Paraphrasing tools can change wording and sentence structure while keeping the original idea, and large language models such as ChatGPT and Gemini can write fluent text that is not copied from any single source, which is now referred to as AI-generated plagiarism [1].

Commercial checkers exist, but they are commonly subscription-based and usually require documents to be sent to external servers, which raises privacy concerns for unpublished student work and puts them out of reach of many polytechnics and colleges. Surveys also report that existing detection tools struggle with AI-generated text and can produce false positives [1], [2]. Evaluators additionally spend considerable time collecting evidence by hand.

This paper presents DeltaDetect, a proposed system that detects direct copying, deep paraphrasing and AI-generated text in one workflow and delivers the result as an automatically generated PDF report. Its main idea is to combine lexical and semantic similarity and to use an offline, self-trained AI-content classifier so that no document leaves the local machine. The AI-content detection module is developed and trained in-house, and the complete system is free to use, with no subscription or licence fee. The paper describes the design of the system; quantitative results will be reported after evaluation is complete.

II. LITERATURE REVIEW

- 1) Pudasaini et al. (2025) surveyed datasets, algorithms, tools and evasion strategies for plagiarism detection in the era of ChatGPT and Gemini, and identified gaps in existing solutions [1]. Makhmutova et al. (2025) surveyed plagiarism types and detection algorithms, contrasting string matching with machine learning and deep learning methods, and discussed cross-

language and AI-generated plagiarism as open challenges [2].

- 2) Hans et al. (2024) proposed Binoculars, a zero-shot method for detecting machine-generated text that contrasts the perplexity of a text under one language model with the cross-perplexity between two related models, so no training data from the target model is needed. They reported that it detects over 90% of machine-generated samples at a false-positive rate of 0.01%, which shows that very low false-positive rates are the target for detectors such as the one in DeltaDetect [3].
- 3) Mitchell et al. (2023) presented DetectGPT, a zero-shot method for detecting machine-generated text. It is based on the observation that text sampled from a language model tends to lie in regions of negative curvature of the model's log-probability function, so no separate classifier or labelled dataset is needed [4].
- 4) Liang et al. (2023) evaluated several widely used GPT detectors and found that they consistently misclassified writing by non-native English speakers as AI-generated. Their work shows the risk of false positives and unfair outcomes, and the need for careful thresholds and human review [5].
- 5) Sadasivan et al. (2023) examined how reliably AI-generated text can be detected and showed, both empirically and theoretically, that paraphrasing can make machine-generated text evade several kinds of detectors, and that detection becomes harder as language models improve. Their findings support combining semantic matching with human review, as DeltaDetect does [6].
- 6) Reimers and Gurevych (2019) introduced Sentence-BERT, which modifies BERT [7] with a siamese network structure to produce sentence embeddings that can be compared with cosine similarity. For finding the most similar pair among 10,000 sentences, this reduced the effort from about 65 hours with BERT to about 5 seconds, making meaning-based matching of paraphrased text practical [9].
- 7) Schleimer, Wilkerson and Aiken (2003) proposed winnowing, a local document fingerprinting algorithm. It selects a small subset of hash values from each document so that any shared passage longer than a guaranteed length is still detected, which makes large-scale detection of verbatim copying efficient [11].

III. PROPOSED SYSTEM

The proposed work involves the design and development of DeltaDetect, a web-based system for plagiarism and AI-content detection. A user uploads a document in PDF, DOCX or TXT format. The system extracts and preprocesses the text, compares it with a reference collection using a weighted combination of TF-IDF lexical similarity and SBERT semantic similarity, and passes each passage to an offline classifier that estimates the probability that it was machine-generated. Finally, the findings are compiled into a structured PDF report.

The system is designed with a React.js and Tailwind CSS frontend, a Django REST Framework backend and an AI/ML pipeline using scikit-learn [10], Sentence-BERT, PyTorch [8] and NLTK, with PyPDF2, python-docx and ReportLab for file parsing and reporting. Because the trained model runs locally, no third-party API is needed, the system is free to use with no subscription or licence fee, and documents stay on the user's system.

IV. EXISTING SYSTEM

Existing plagiarism checkers mainly compare submitted text with databases of known sources using keyword and string matching. These methods find verbatim and lightly edited copying well, but their accuracy falls when words are replaced by synonyms or sentences are reordered, so deeply paraphrased content is often missed. Tools that add AI-text detection are typically offered as online services, which require documents to be uploaded to third-party servers and are commonly provided on a paid basis. Studies have also shown that some AI-text detectors wrongly flag the writing of non-native English speakers as machine-generated [5]. Evaluators must still collect evidence and prepare reports manually. The existing system therefore needs an affordable, private and more accurate approach that combines lexical matching, semantic matching and AI-content detection with automated reporting.

V. SYSTEM ARCHITECTURE

DeltaDetect follows a layered architecture (Fig. 1). The presentation layer is a web interface for uploading documents and viewing results. The application layer,

built with Django REST Framework, handles requests, authentication and job orchestration. The AI/ML layer contains the TF-IDF component, the Sentence-BERT model, the PyTorch AI-content classifier and NLTK preprocessing. The data and output layer stores users, documents and results in SQLite or MongoDB and generates the PDF reports with ReportLab.

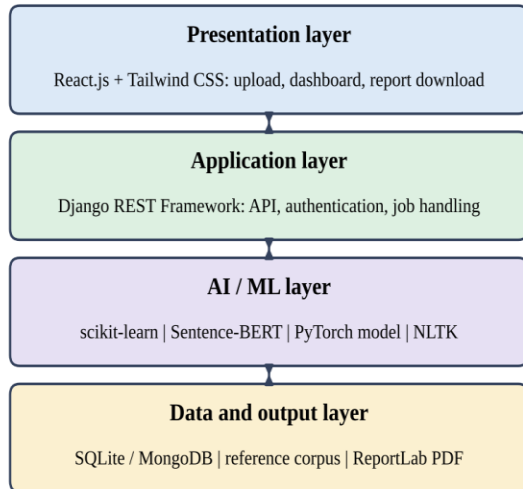


Fig. 1: System Architecture

VI. BLOCK DIAGRAM

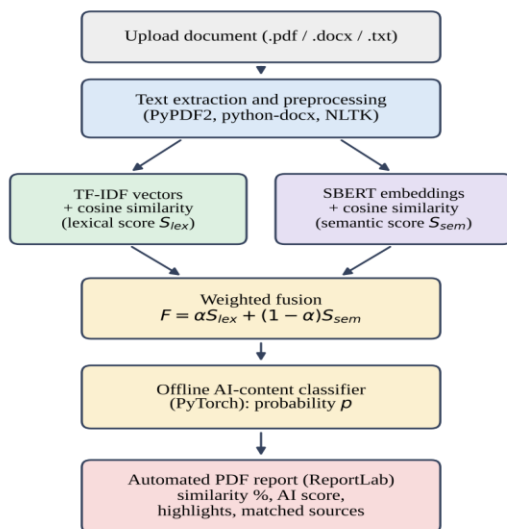


Fig. 2: Block diagram

- 1) Preprocessing: The uploaded file is converted to text with PyPDF2 or python-docx, cleaned, split into sentences and words with NLTK, lower-cased, and stripped of stop-words for the lexical branch.

- 2) TF-IDF Branch: The text is converted to weighted TF-IDF vectors and compared with the reference documents using cosine similarity, which gives the lexical score.
- 3) SBERT Branch: Each sentence is converted to a Sentence-BERT embedding and compared with the reference embeddings using cosine similarity, which gives the semantic score and exposes paraphrased content.
- 4) AI-Content Classifier: A locally trained PyTorch model estimates, for each passage, the probability that it is machine-generated.
- 5) Report Generator: ReportLab compiles the similarity percentage, AI-content score, highlighted passages and matched sources into a PDF report.

VII. WORKING PRINCIPLE OF DELTADTECT

DeltaDetect works on a compare-and-classify principle. For the lexical branch, a term t in document d from a collection of N documents receives the TF-IDF weight

$$\text{tfidf}(t,d) = \text{tf}(t,d) \cdot \log(N / \text{df}(t)) \quad (1)$$

where $\text{df}(t)$ is the number of documents that contain t . Two vectors A and B are compared by cosine similarity,

$$\cos(A,B) = (A \cdot B) / (|A| |B|) \quad (2)$$

which lies between 0 and 1 for TF-IDF vectors and does not depend on document length. The same measure is applied to SBERT embeddings. The lexical score S_{lex} and the semantic score S_{sem} are combined into one similarity value,

$$F = \alpha \cdot S_{\text{lex}} + (1 - \alpha) \cdot S_{\text{sem}} \quad (3)$$

where α between 0 and 1 is a weight tuned on a validation set. In parallel, the classifier outputs a probability p for each passage, and passages above a tuned threshold are flagged as likely AI-generated. This probability is an indicator, not proof, and the final decision is left to the human reviewer. The weighted score and threshold tuning are the planned means of keeping false positives low on paraphrased text.

VIII. MODULES

- 1) USER (STUDENT / TEACHER):
 - Login.
 - Upload document.

- View similarity and AI-content result.
- Download PDF report.
- 2) PREPROCESSING MODULE:
 - Extract text from PDF, DOCX and TXT files.
 - Clean, tokenise and remove stop-words.
- 3) SIMILARITY ENGINE:
 - Build TF-IDF vectors and compute cosine similarity.
 - Create SBERT embeddings and compute semantic similarity.
 - Fuse the scores into a similarity percentage.
- 4) AI-CONTENT DETECTOR:
 - Load the offline trained model.
 - Classify passages as human-written or AI-generated.
- 5) REPORT GENERATOR:
 - Highlight flagged passages and matched sources.
 - Compile and save the PDF report.
- 6) ADMIN:
 - Manage users and the reference collection.
 - Update and retrain the AI-content model.

IX. ADVANTAGES

- 1) Detects copying, paraphrasing and AI-written text in one workflow.
- 2) Understands meaning through SBERT, so rewritten content is matched.
- 3) Runs offline, so documents stay private and no internet connection is needed for analysis.
- 4) Uses open-source tools, so there are no licence fees.
- 5) Produces a consistent, automated PDF report that saves evaluators' time.
- 6) Modular design allows each component to be improved independently.

X. LIMITATIONS AND FUTURE SCOPE

The AI-content model needs diverse, good-quality training data and continuous fine-tuning, and its output is probabilistic and may degrade when AI text is heavily edited or new AI models appear. Transformer models need more compute than keyword methods, so large documents may be slow, and matching is limited to the reference collection available to the system. Results must therefore be reviewed by a human and are not proof of misconduct.

The system will be evaluated on original, copied, paraphrased and AI-generated test documents using precision, recall, F1-score and false-positive rate, with the weight α and thresholds tuned on a separate validation set. Future work includes regional Indian languages, cross-language and source-code plagiarism, integration with learning-management systems and explanations of why a passage was flagged.

XI. CONCLUSION

This paper presented DeltaDetect, a hybrid TF-IDF and SBERT plagiarism detector with an offline AI-content classifier and automatic PDF reporting. TF-IDF is fast and explainable but cannot see through paraphrasing, whereas SBERT understands meaning but is heavier and less transparent; combining them, together with a locally trained classifier, addresses the main weaknesses of current tools while staying private and free to run. Since such systems give probabilistic evidence, careful threshold tuning and human review remain essential. DeltaDetect is intended to support teachers with clear, explainable evidence and to encourage honest, original work.

REFERENCES

- [1] Pudasaini, S., Miralles-Pechuán, L., Lillis, D., and Llorens Salvador, M., "Survey on AI-Generated Plagiarism Detection: The Impact of Large Language Models on Academic Integrity", *Journal of Academic Ethics*, vol. 23, no. 3, pp. 1137–1170, 2025.
- [2] Makhmutova, A., Turan, C., and Amirzhanov, A., "Plagiarism Types and Detection Methods: A Systematic Survey of Algorithms in Text Analysis", *Frontiers in Computer Science*, 2025.
- [3] Hans, A., Schwarzschild, A., Cherepanova, V., Kazemi, H., Saha, A., Goldblum, M., Geiping, J., and Goldstein, T., "Spotting LLMs With Binoculars: Zero-Shot Detection of Machine-Generated Text", *Proc. International Conference on Machine Learning (ICML)*, PMLR 235, 2024.
- [4] Mitchell, E., Lee, Y., Khazatsky, A., Manning, C. D., and Finn, C., "DetectGPT: Zero-Shot Machine-Generated Text Detection using Probability Curvature", *Proc. International Conference on Machine Learning (ICML)*, 2023.

- [5] Liang, W., Yuksekgonul, M., Mao, Y., Wu, E., and Zou, J., “GPT Detectors are Biased against Non-Native English Writers”, *Patterns*, vol. 4, no. 7, 2023.
- [6] Sadasivan, V. S., Kumar, A., Balasubramanian, S., Wang, W., and Feizi, S., “Can AI-Generated Text be Reliably Detected?”, *arXiv preprint arXiv:2303.11156*, 2023.
- [7] Devlin, J., Chang, M.-W., Lee, K., and Toutanova, K., “BERT: Pre-training of Deep Bidirectional Transformers for Language Understanding”, *Proc. NAACL-HLT*, 2019, pp. 4171–4186.
- [8] Paszke, A., et al., “PyTorch: An Imperative Style, High-Performance Deep Learning Library”, *Advances in Neural Information Processing Systems* 32, 2019.
- [9] Reimers, N., and Gurevych, I., “Sentence-BERT: Sentence Embeddings using Siamese BERT-Networks”, *Proc. EMNLP-IJCNLP*, 2019, pp. 3982–3992.
- [10] Pedregosa, F., et al., “Scikit-learn: Machine Learning in Python”, *Journal of Machine Learning Research*, vol. 12, pp. 2825–2830, 2011.
- [11] Schleimer, S., Wilkerson, D. S., and Aiken, A., “Winnowing: Local Algorithms for Document Fingerprinting”, *Proc. ACM SIGMOD International Conference on Management of Data*, 2003, pp. 76–85.