SmartPlag: Academic Plagiarism Detection with Multilingual Model and Code Analysis

Dr. Gajanan Arsalwad
(Guide)
Department of Information Technology, Savitribai Phule Pune University, Pune, India
Email*: gajananarsalwad.tcoer@kjei.edu.in*

Vrushali Gawai
Department of Information Technology, Savitribai Phule Pune university, Pune, India,
vrushalig2004@gmail.com

Sanika Deshmukh
Department of Information Technology, Savitribai Phule Pune university, Pune, India,
sanikadeshmukh5707@gmail.com

Trupti Dhandar
Department of Information Technology, Savitribai Phule Pune university, Pune, India,
trupti.dhandar64@gmail.com

Abstract

Generative AI has made plagiarism detection more complex, as Large Language Model (LLMs) is now able to create essays and generate code very efficiently. This survey examines the plagiarism detection in natural language, multilingual low resource contexts and source code. For that We have reviewed benchmarks like CodeMirage [1], obfuscation-resistant methods [2], fuzzy severity models [3], stylometric and token-based techniques [4][5], Graph based models [8], and behavioural similarity measures [10]. From that we have key findings of remaining gaps: multilingual detection (Hindi, Marathi), difficulty identifying logic preserving code plagiarism and faculty friendly systems. By combining these the survey outlines -robust multilingual benchmarks, semantics- aware code analysis and transparent hybrid frameworks- for building AI resistant plagiarism detection.

Introduction

Plagiarism detection is a challenge in academia, weakening the integrity of education and research. The wide availability of online resources has made copying easier that has violated the traditional detection methods. That are focused only on string similarity and lexical overlap often fail against paraphrased and restructured content. The rise of generative AI models such as ChatGPT and Codex has increased the issue, that enables student and researchers to produce rewritten text and code that bypass conventional tools.

Recent research has searched these gaps through semantic code analysis, structural approaches [3][5], and linguistic models for adapted to low resource languages like Hindi and Marathi [4]. Benchmarks such as CodeMirage [1], highlight the need for systems robust to AI adversarial manipulation and adaptable to multilingual and multimodal context. Our survey aims to meet this need by integrating semantic code analysis, multilingual adaptability and explainable AI insights. The framework provides comprehensive solution to plagiarism detection and code level obfuscation, with paper structured review literature, outline methodology, present findings and discuss contribution and future directions.

Literature Survey

The problem of plagiarism detection has been extensively studied across domains of text, code, and multimodal data, with the emergence of AI-generated content presenting new challenges. Traditional plagiarism detection approaches relied heavily on string matching and lexical overlap, but these quickly fall short when students or professionals adopt paraphrasing techniques or leverage LLMs (Large Language Models) to rephrase and generate unique-looking outputs.

Recent works have introduced benchmark datasets to evaluate plagiarism and AI-generated content detection systems. For instance, the CodeMirage dataset [1] focuses on multilingual AI-generated code and emphasizes the importance of semantic evaluation of programming logic rather than surface-level textual comparison. This is highly relevant to our proposed system, which seeks to detect plagiarism even when code has been restructured or paraphrased using AI tools. Similarly, shared tasks such as PAN 2025 [2] provide large-scale benchmarks in detecting AI-generated text, proving the necessity of robust stylistic and semantic analysis.

Reference	Core Functions	Technologies & Control	Key Features
[1] CodeMirage (2025)	Detects AI-generated & paraphrased multilingual code	Multilingual benchmark dataset; semantic evaluation	First dataset for **AI-generated/paraphrased code plagiarism**; focuses on **semantic logic**
[2] PAN 2025 Overview (2025)	Generative AI plagiarism & writing style analysis	Shared tasks; multilingual evaluation	Large-scale **benchmark for AI plagiarism detection**; supports **cross-language and stylistic analysis**
[3] Mutsaddi & Choudhary (2025)- Enhancing plagiarism detection in Marathi with a weighted ensemble of TF-IDF and BERT embeddings for low-resource language processing	Plagiarism detection in Marathi	TF-IDF + BERT embeddings; ensemble methods	Addresses **low-resource languages**; improves detection in **regional academic texts**
[4] Sharmila et al. (2024) – *FTLM*	Severity assessment of plagiarism	Fuzzy TOPSIS + language modelling	Goes beyond detection to **assess severity** of plagiarism
[5] Parvathy et al. (2024)- Automated Code Assessment and Feedback: A Comprehensive Model for Improved Programming Education	Automated plagiarism detection in programming education	ML pipeline; automated assessment & feedback	Integrates plagiarism detection with **feedback for learners**
[6] Sağlam et al. (2024)- Detecting Automatic Software Plagiarism via Token Sequence Normalization.	Software plagiarism detection via normalization	Token sequence normalization + ML classifiers	Resists **code obfuscation** by normalizing syntax
[7] Cheers et al. (2023) – *SPPlagiarise*	Simulated semantics-preserving plagiarism in Java	Code transformation; semantic embeddings	Provides **benchmark for semantic plagiarism**
[8] Manzoor et al. (2023) - Exploring the Landscape of Intrinsic Plagiarism Detection: Benchmarks, Techniques, Evolution, and Challenges	Intrinsic plagiarism detection survey	Comparative analysis of benchmarks & techniques	Maps **evolution of plagiarism detection**; highlights **gaps & challenges**

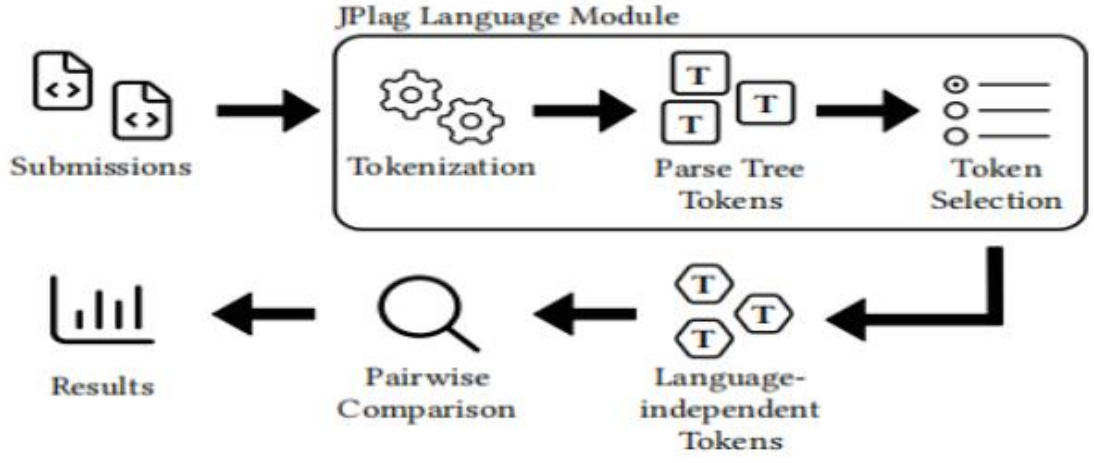

Fig. Token Based Plagiarism Detection Using JPlag [17]

Previous techniques used for check Plagiarism:

Ref No.	Techniques Used	Limitations
[1] CodeMirage Benchmark	Multilingual code benchmark; Evaluates detectors with embeddings and classifiers	Benchmark-focused; lacks institutional/classroom datasets; limited regional-language analysis
[2] TF-IDF + BERT Ensemble	Weighted ensemble for Marathi text plagiarism	Focused on text only; weak on code plagiarism and AI-generated paraphrases
[3] PAN 2025 Tasks	Generative AI detection; multilingual style analysis	English-centric; limited focus on code
[4] Semantic Similarity Search	Code embeddings, retrieval-based similarity	Sensitive to obfuscation and structural edits
[5] Blockchain Provenance	Blockchain for submission authenticity and originality proof	High overhead; complexity for institutional use
[6] General String Matching	Exact word/phrase comparison	Cannot detect paraphrases; fails for semantics
[7] TF-IDF Vectorization	Statistical word-weighting	Sensitive to vocabulary; struggles with paraphrase
[8] Cosine Similarity	Vector-based similarity score	Works for exact matches; weak deep semantic matching
[9] N-gram Analysis	Sequential chunk comparison	High cost; poor paraphrase detection
[10] Basic ML Models	SVM, Random Forest on handcrafted features	Needs heavy feature engineering; weak adaptability
[11] Cross-Language Matching	Translate + match	Translation errors reduce accuracy

Challenges

[1] SmartPlag faces several challenges in its implementation. One major challenge is detecting semantic paraphrasing (**when the wording of a document changes but the meaning stays the same**).

[2] Even advanced NLP models sometimes struggle with such subtle rewording. Another challenge is detecting AI-generated content, which is becoming a growing concern in plagiarism detection and remains a developing research area.

[3] Scalability is also a critical issue, as maintaining and searching through large document repositories requires significant storage and high-performance indexing systems.

[4] Multilingual detection is another hurdle, as cross-language plagiarism requires sophisticated translation models and can suffer from translation inaccuracies.

[5] Data privacy and security are crucial, since academic submissions often contain sensitive information, and storing them securely while ensuring accessibility must comply with legal requirements.

[6] Finally, computational cost is a challenge, as embedding generation and similarity searches require substantial computing resources, especially for real-time plagiarism analysis, and the quality of the reference database is fundamental to accuracy but difficult to maintain due to document diversity and format variation.

Some Research Questions:

Domain	Research Questions	Motivation
Plagiarism Detection	How can we accurately detect plagiarism in academic documents using AI-based techniques?	To ensure originality of academic submissions and prevent intellectual property theft.
Natural Language Processing (NLP)	What NLP algorithms and embeddings provide the best semantic similarity detection for text comparison?	To improve detection accuracy beyond exact-match algorithms, enabling detection of paraphrased plagiarism.
Machine Learning & AI	How can we train models to differentiate between genuine academic writing and AI-generated or paraphrased text?	To strengthen plagiarism detection against sophisticated content generation tools.
Database & Indexing	How should a large repository of academic documents be stored and indexed for fast plagiarism checking?	To optimize performance and scalability for real-time plagiarism detection across large datasets.
User Interface & Experience	How can plagiarism reports be presented in a clear and actionable format to users?	To make the plagiarism detection tool user-friendly for students, teachers, and admins.
Security & Privacy	How can document data be stored securely while ensuring compliance with privacy regulations?	To protect sensitive academic content and comply with data protection laws.
Multilingual Plagiarism Detection	How can we extend plagiarism detection to work for documents in multiple languages?	To broaden applicability for global academic environments.
AI-Generated Content Detection	How effective are current AI content detection methods, and how can they be integrated into SmartPlag?	To prevent misuse of AI tools for generating plagiarized academic work and improve trust in academic integrity systems.

Proposed Solution:

Our proposed solution, **SmartPlag**, is an academic plagiarism detection system engineered to overcome the limitations of traditional, lexical-based tools. It employs a two-part architecture that combines an AI-powered detection engine with a dedicated web crawler, enabling it to accurately identify plagiarism from both translated documents and source code. The complete system workflow is represented in the following architecture diagram

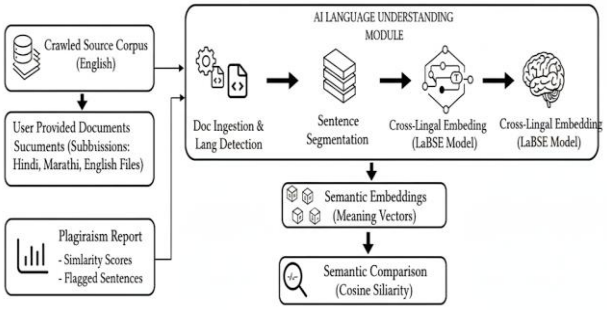

Fig. Cross-Lingual Semantic Plagiarism Detector Architecture

The system architecture is designed as a modular pipeline that processes documents in a series of logical stages. It begins with the ingestion of documents, followed by sophisticated AI-powered language understanding, and concludes with a comprehensive plagiarism report.

- **Document Ingestion:** The system accepts suspicious documents and an English source corpus. This corpus can be either manually provided or automatically built by a dedicated web crawler.
- **AI Language Understanding Module:** This is the core of our solution. It takes the textual data and performs several key steps:
    - **Document Ingestion & Language Detection:** The system reads the documents from various file formats (.txt, .pdf, .docx) and identifies their language.

- **Sentence Segmentation:** The text is broken down into individual, analyzable sentences.
- **Cross-Lingual Embedding (LaBSE Model):** This is the key AI technique. The AI model converts each sentence into a unique numerical vector that represents its meaning. This process allows the system to understand that a sentence in Hindi and a sentence in English can have the same meaning.

- **Semantic Comparison:** The numerical vectors are then compared using **Cosine Similarity**, a mathematical technique that measures the "closeness" of the vectors. The closer the vectors are, the more semantically similar the sentences are.
- **Plagiarism Report:** Based on the similarity scores, the system generates a final report, flagging sentences that exceed a set plagiarism threshold.

Conclusion:

We built a smart plagiarism checker that solves a problem no ordinary tool can. We all know people can copy-paste from the internet, but what if they translate it first? Our system is designed to catch exactly that.

Instead of just matching words, our tool's AI "brain" understands the meaning of what's written. This means it can read a paper in Hindi and compare the ideas in it to what it has found on English websites.

And we didn't stop there. Our system is built to handle both written papers and computer code. It's a two-part system: a "web scout" goes out and collects information from the internet, and a "checker" then quickly and smartly finds any matching content in your document.

So, in the end, we've created a working blueprint for a powerful tool that can find hidden plagiarism and help keep things fair and honest in academics.

References:

[1] **May 2025 -** CodeMirage: A Multi-Lingual Benchmark for Detecting AI-Generated and Paraphrased Source Code from Production-Level LLMs.

[2] J. Bevendorff et al., "Overview of PAN 2025: Generative AI Detection, Multilingual Text Detoxification, Multi-Author Writing Style Analysis, and Generative Plagiarism Detection," ECIR/PAN 2025.

[3] Mutsaddi, A., & Choudhary, A. P. (2025). Enhancing plagiarism detection in Marathi with a weighted ensemble of TF-IDF and BERT embeddings for low-resource language processing. arXiv. https://arxiv.org/abs/2501.05260

[4] **2024** - P. Sharmila, Kalaiarasi Sonai Muthu Anbananthen, Nithyakala Gunasekaran, Baarathi Balasubramaniam, and Deisy Chelliah. "FTLM: A Fuzzy TOPSIS Language Modeling Approach for Plagiarism Severity Assessment."

[5] **2024** - R. Parvathy, M. G. Thushara, and Jinesh M. Kannimoola. "Automated Code Assessment and Feedback: A Comprehensive Model for Improved Programming Education."

[6] **2024** - Timur Sağlam, Moritz Brödel, Larissa Schmid, Sebastian Hahner. "Detecting Automatic Software Plagiarism via Token Sequence Normalization."

[7] **2023** - Hayden Cheers, Yuqing Lin, and Shamus P. Smith. "SPPlagiarise: A tool for generating simulated semantics-preserving plagiarism of java source code."

[8] **2023** - Muhammad Faraz Manzoor, Muhammad Shoaib Farooq, Muhammad Haseeb, Uzma Farooq, Sohail Khalid, and Adnan Abid. "Exploring the Landscape of Intrinsic Plagiarism Detection: Benchmarks, Techniques, Evolution, and Challenges."

[9] **2023** - Hayden Cheers and Yuqing Lin. "A novel graph-based program representation for java code plagiarism detection."

[10] **2023** - M. Duraçík, E. Krşák, and P. Hrkút. "Current trends in source code analysis, plagiarism detection and issues of analysis big datasets."

[11] **2021** - Hayden Cheers, Yuqing Lin, and Shamus P. Smith. "Academic Source Code Plagiarism Detection by Measuring Program Behavioral Similarity."

[12] **2020** - Michal Duracik, Patrik Hrkut, Emil Krsak, and Stefan Toth. "Abstract Syntax Tree Based Source Code Antiplagiarism System for Large Projects Set."

[13] **2020** - Breanna Devore-McDonald and Emery D. Berger. "Mossad: Defeating Software Plagiarism Detection."

[14] **2019** - M. Duraçík, E. Kršák, and P. Hrkút. "Searching source code fragments using incremental clustering."

[15] **2019** - J. Martínez-Gil, "Source code clone detection using unsupervised similarity measures,"

[16] **2024** - Timur Sağlam, Sebastian Hahner, Larissa Schmid, Erik Burger. "Automated Detection of Al-Obfuscated Plagiarism in Modeling Assignments."

[17] **2024-**IEEE/ACM 46th International Conference on Software Engineering (ICSE) - Detecting Automatic Software Plagiarism via Token Sequence Normalization