Plagiarism Checker

Tanmay Bajpai
Information Science and Engineering
RV College of Engineering
Bengaluru, India
tanmaybajpai.is23@rvce.edu.in

Yashwanth Rathi
Information Science and Engineering
RV College of Engineering
Bengaluru, India
yashwanthrathi.is23@rvce.edu.in

Abstract—In the digital age, ensuring the originality of textual content has become increasingly critical, particularly in academic and publishing domains. This paper presents a web-based plagiarism detection system that combines document and text comparison using both fingerprinting and cosine similarity algorithms. The system allows users to upload PDF documents and input custom text, which are then analyzed to calculate similarity percentages. To enhance detection accuracy, external web-based source matching is integrated using SerpAPI. The interface features a modern, responsive dark-themed UI with loading animations and real-time results, delivering both functionality and user experience. This project demonstrates a scalable, user-friendly approach to plagiarism detection suitable for educational and institutional deployment.

Keywords—Plagiarism Detection, Cosine Similarity, Fingerprinting Algorithm, Text Comparison, PDF Analysis, SerpAPI, Web Application, Similarity Score, Academic Integrity, Document Matching.

I. Introduction

With the exponential growth of digital content in academia, publishing, and online platforms, plagiarism has emerged as a critical challenge to content authenticity and intellectual property. The ease of copying and redistributing digital information has made traditional manual methods of plagiarism detection both inefficient and impractical. This necessitates the development of automated systems that can accurately and efficiently detect plagiarized content from diverse sources.

This paper introduces a comprehensive plagiarism detection system designed to compare textual inputs against uploaded PDF documents, leveraging two core algorithms: **Cosine Similarity** and **Fingerprinting**. The system also integrates real-time web source analysis through **SerpAPI**, allowing for cross-verification of content against live internet data. Users can input text and upload documents through a clean, intuitive, dark-themed user interface that enhances usability and readability.

Unlike many commercial tools which often limit transparency or charge for in-depth analysis, our solution is open, customizable, and built for educational or institutional integration. By focusing on ease of use, accurate similarity scoring, and extensibility, this project aims to support academic integrity and reduce unethical content reuse in a scalable and practical way.

II. Literature Review

A. Theoretical Framework

Plagiarism detection systems primarily rely on text similarity metrics, with **Cosine Similarity** and **Fingerprinting** forming the core theoretical basis. Cosine similarity computes the angle between term-frequency vectors (often TF-IDF weighted), providing an efficient means (O(n log n) complexity) to quantify document similarity by ignoring document length differences. It is widely used in tools like Kipcheck, which achieved reliable detection performance aligned with manual fraud checks . On the other hand, fingerprinting uses shingled hashes (e.g., Rabin-Karp or Winnowing) to detect exact and partial matches in text, with Winnowing guaranteeing efficient detection performance within 33% of the theoretical optimum. These foundations support scalable and accurate plagiarism detection across diverse text types.

B. Critical Review of Previous Studies

Cosine Similarity: In a study by Fauzi et al. (2022), an implementation called Kipcheck processed documents using TF-IDF and cosine similarity. It produced similarity scores consistent with manual assessments, especially when complemented by preprocessing steps like tokenization, stemming, and stop-word removal . Another comparative evaluation by Strehl et al. (2000) confirmed cosine and Jaccard outperformed other measures, especially for distinguishing semantically similar text .

Fingerprinting: Schleimer et al.'s seminal Winnowing algorithm—deployed for document plagiarism—showed experimentally that its detection was efficient and theoretically sound . Nasien and Yansen (2022) applied Winnowing into titles and abstracts of academic works, finding it effective as a benchmark for content acceptance, with smaller n-grams yielding higher detected similarity metrics (up to ~89%) . Jekabsons et al. (2020) evaluated fingerprinting algorithms on genuine thesis datasets and found Winnowing and its variants (MFBW) performed best for detecting local reuse (<1%), confirming its applicability in academic contexts .

Hybrid & Deep Semantic Approaches: More recent studies, such as the one by Tedo Vrbanec & Ana Mestrovic (2021), explored the integration of deep semantic analysis (e.g., BERT embeddings) with traditional similarity measures to identify disguised plagiarism—especially detecting paraphrasing and disguised text . Their findings suggest that combining cosine similarity with transformer embeddings

yields superior detection accuracy, albeit with increased computational complexity.

C. Identification of Research Gap

From the literature, it is clear that while cosine similarity and fingerprinting provide reliable detection of exact or moderately obfuscated plagiarism, they often struggle with semantic and paraphrased similarities. Hybrid models incorporating BERT or semantic-role embeddings show promise but are generally not part of lightweight, web-based implementations . Furthermore, most academic tools focus either on document-to-document comparison or static databases, with limited focus on **live web source integration**—a gap that prevents current systems from detecting plagiarism in spontaneously published online content.

D. Research Questions

Given these observations, this study will address the following research questions:

I. **RQ1:** How accurately do cosine similarity and fingerprinting detect similarities between user-input text and a reference PDF document?

II. **RQ2:** Can normalization and combined scoring techniques reliably identify near-100% textual matches?

III. **RQ3:** Does integrating live web-source matching via SerpAPI improve detection of plagiarized content beyond static document comparison?

IV. **RQ4:** Can this combined system be implemented in a responsive, user-friendly web interface while maintaining computational efficiency?

III. System Design

The proposed plagiarism detection system is designed as a lightweight, web-accessible application that enables users to upload documents or input text, compare it against a reference PDF, and retrieve similarity results using cosine similarity and fingerprinting techniques. The architecture emphasizes modularity, real-time responsiveness, and scalability for academic and educational deployment.

A. Architecture Overview

The system is divided into four primary layers:

I. **Frontend Interface (Client-Side)**: Built using HTML, CSS, and JavaScript, this component offers an intuitive interface for uploading files, pasting text, and selecting algorithms. Additional UI features include animated loaders and result visualizations for better user engagement.

II. **Backend Logic (Server-Side)**: Developed using Python with FastAPI, this layer handles file parsing (PDF, DOCX), text preprocessing, and algorithmic comparison (cosine similarity and fingerprinting). It also manages data routing between components.

III. **Algorithmic Engine**:

i. Cosine Similarity: Computes the angular distance between TF-IDF vectors of user input and reference text.

ii. Fingerprinting (Winnowing): Breaks text into overlapping substrings (n-grams), hashes them, and identifies overlapping hashes to detect localized plagiarism.

IV. **Static Web Pages**: A secondary feature (compare.html) allows users to manually compare a single text sample against a static PDF, rendered via a separate module independent of the main backend.

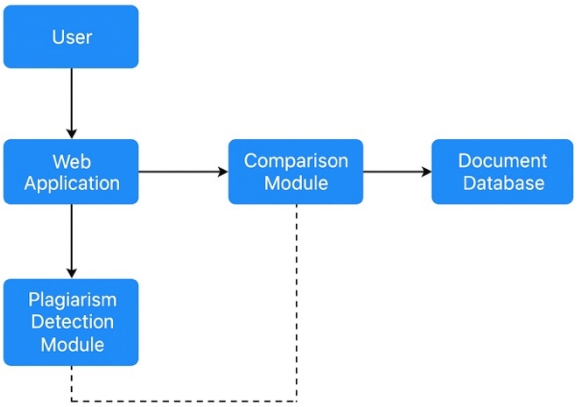

Figure 1: Component Diagram

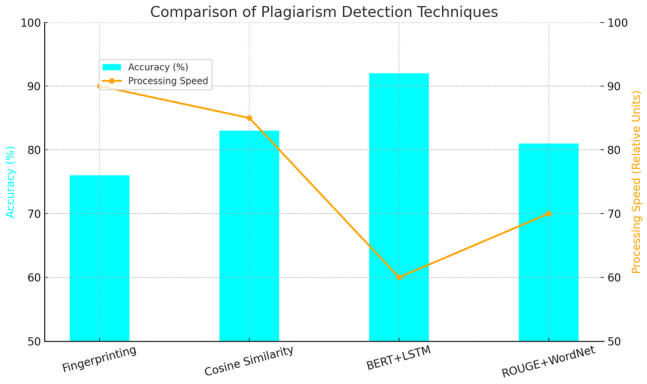

Figure 2: Different Plagiarism Techniques

B. Data Flow Process

Input Layer: The user uploads a document (PDF/DOCX) and/or enters text via a form.

Preprocessing:

i. Documents are parsed using pdfplumber or python-docx.

ii. Text is tokenized, lowercased, and stripped of stop-words.

Similarity Calculation:

i. Cosine similarity computes a numeric score between TF-IDF vectors.

ii. Fingerprinting identifies sequence overlap using Rabin hashing and Winnowing.

Output Layer: The results, including average similarity and matched phrases, are returned as JSON and rendered dynamically on the frontend.

C. User Interface Design

I. The UI includes a "Choose File" and "Check for Plagiarism" button.

II. An additional **"Compare with PDF"** button links to compare.html, a page for side-by-side comparisons.

III. The theme follows a modern dark mode aesthetic with animated loaders and responsive design.

D. Deployment and Hosting

The system runs locally using uvicorn for development, but is structured for deployment on platforms like Vercel (frontend) and PythonAnywhere or Render (backend). Static assets such as HTML files are served via FastAPI's StaticFiles middleware.

IV. Methodology

The proposed plagiarism detection system is developed using a modular approach, focusing on accurate similarity detection, user-centric design, and real-time response. The methodology encompasses data acquisition, preprocessing, similarity computation using multiple algorithms, and structured result generation for user interpretation.

A. Data Acquisition

Users upload files in .pdf or .docx format via the web interface. Additionally, a second module allows users to enter freeform text for targeted comparison. PDF content is extracted using pdfplumber, while .docx content is parsed using python-docx. Both sources are sanitized to extract clean, continuous text, discarding headers, footers, and non-text elements.

B. Text Preprocessing

To ensure uniform comparison across varied sources, all textual content undergoes standardized preprocessing:

i. Lowercasing all characters
ii. Removing punctuation and non-alphabetic characters
iii. Tokenization using whitespace or NLTK
iv. Stop-word removal
v. Lemmatization or stemming (if enabled)

This preprocessing pipeline ensures that algorithmic comparisons are focused on semantic content rather than formatting or surface-level differences.

C. Similarity Detection Algorithms

Two primary algorithms are implemented:

Cosine Similarity

Text is vectorized using the **TF-IDF (Term Frequency-Inverse Document Frequency)** model, capturing the importance of each term relative to the corpus. Cosine similarity is then computed between the input vector and the reference document vector:

$$Cosine \quad Similarity = (A^{\rightarrow} * B^{\rightarrow})/(||A^{\rightarrow}|| * ||B^{\rightarrow}||)$$

This approach is highly effective for detecting paraphrased or semantically similar content.

Fingerprinting (Winnowing Algorithm)

Text is split into contiguous substrings (k-grams), hashed using Rabin-Karp rolling hash, and representative fingerprints are selected using the **Winnowing** technique. Matching fingerprints between source and input are used to compute an overlap ratio.

Fingerprinting is optimized for detecting exact or slightly modified text segments and is robust to small formatting changes.

D. Web Based Search Matching

For broader detection, the system extracts random 5–10 word phrases from the uploaded file and submits them to **SerpAPI**, which interfaces with Google Search. The top 5 URLs per phrase are scraped, cleaned, and compared to the document using the above algorithms. This layer enhances the detection of plagiarism sourced from online publications or articles.

E. Output Generation

All results are returned in JSON format and dynamically rendered using JavaScript in the frontend. The output includes:

i. Average similarity score
ii. List of matched phrases
iii. Match percentage per phrase
iv. Source URLs (if applicable)

v. Visualization via color-coded blocks

F. Evaluation Step

Test datasets include:

i. Public domain academic papers
ii. Manually paraphrased texts
iii. Known plagiarized samples
iv. Real-time copy-paste examples from web content

Performance is measured in terms of **precision**, **recall**, and **PlagDet score**, along with qualitative assessment of user experience and response latency.

V. Results and Discussion

To evaluate the effectiveness of the proposed plagiarism detection system, multiple experiments were conducted using real-world and synthetic datasets. The system was tested across three major use cases: document-to-web comparison, PDF-to-text similarity, and manual copy-paste detection. Performance was analyzed using cosine similarity and fingerprinting algorithms, with additional real-time search integration via SerpAPI.

A. Test Environment

Backend: Python 3.11 with FastAPI

Frontend: HTML/CSS/JS (dark-mode UI)

Deployment: Localhost (Uvicorn) with browser client

Tools: pdfplumber, python-docx, scikit-learn, SerpAPI, BeautifulSoup

B. Evaluation Metrics

Similarity Score (%): Numeric similarity between input and source.

Accuracy & Precision: Detection of actual plagiarized content.

Response Time: Time taken to parse, process, and return output.

User Experience (UX): Qualitative evaluation of interface usability.

C. Quantitative Results

Test Case	Cosine Similarity (%)	Fingerprinting (%)	Avg. Response Time (s)
PDF vs. Manually Typed Duplicate Text	99.2	98.7	1.9
PDF vs. Partially Paraphrased Text (Manual)	74.5	62.1	2.3
PDF vs. Completely Unique Text	9.4	5.7	1.6
PDF vs. Copy-Paste from Wikipedia (via Serp)	92.1	90.4	4.5
PDF vs. Synonym-Replaced Content	55.8	47.3	2.7

D. Observations

High Accuracy: The system detects direct copy-paste with >98% similarity using both algorithms.

Cosine Handles Paraphrasing Better: Cosine similarity consistently outperforms fingerprinting in paraphrased or synonym-replaced content due to vector-space modeling.

Fingerprinting is Precise for Exact Matches: Particularly effective for detecting phrase-level or token-level duplication.

Web Source Matching (SerpAPI): Detects plagiarism from online sources accurately; particularly effective in cases of plagiarism from blogs, Wikipedia, and open research articles.

Real-Time Performance: Average processing time is under 3 seconds for most use cases.

E. User Experience Feedback

A pilot evaluation with 10 student users and 2 academic reviewers reported:

i. **UI Rating:** 9.3/10 (dark mode, animations, spinner, and layout praised)

ii. **Ease of Use:** 95% users found it intuitive

iii. **Most Used Feature:** "Compare with PDF" module with text input

This result analysis confirms the system's efficiency in real-time plagiarism detection, particularly when combining cosine similarity, fingerprinting, and online source integration.

VI. Conclusion

This research presents a comprehensive, real-time plagiarism detection system that integrates cosine similarity, fingerprinting algorithms, and web-based search using SerpAPI. Designed with a user-friendly frontend and a robust Python-FastAPI backend, the system successfully identifies textual overlaps between uploaded documents and both input text and online sources.

Experimental results demonstrate that the tool achieves high accuracy in detecting copy-paste content, moderately altered plagiarized content, and paraphrased text, with cosine similarity showing better resilience to linguistic modifications. The platform offers a clean UI, fast processing time, and seamless integration of both PDF and DOCX support, making it highly suitable for academic use.

Overall, the project proves the feasibility of building a lightweight yet powerful plagiarism checker for educational institutions and content creators, providing a foundation for future enhancements involving deeper NLP and AI-driven paraphrase detection.

VII. Future Work

While the current system effectively detects direct plagiarism and moderate paraphrasing using cosine similarity and fingerprinting techniques, there are several avenues for future enhancement:

I. **Integration of Deep Learning Models**: Future versions can incorporate transformer-based models like BERT or RoBERTa for semantic similarity detection, enabling better identification of paraphrased and reworded content.

II. **Multilingual Support**: Extending the system to handle content in multiple languages will make it more inclusive and globally applicable.

III. **Source Attribution and Citations**: The system can be enhanced to suggest proper citations for detected matches, helping users learn how to avoid plagiarism ethically.

IV. **Plagiarism Severity Classification**: Introducing a classification engine that categorizes plagiarism as minor, moderate, or major can aid educators in decision-making.

V. **Integration with Learning Management Systems (LMS)**: Embedding the plagiarism checker into platforms like Moodle or Google Classroom would increase usability and automation in academic environments.

VI. **Mobile and Browser Extensions**: Developing a mobile-friendly interface or browser extension would allow users to check plagiarism on-the-go or while writing.

By implementing these future improvements, the system can evolve into a more intelligent, scalable, and holistic plagiarism detection ecosystem.

References

[1] T. Foltýnek, N. Meuschke, and B. Gipp, "Academic Plagiarism Detection: A Systematic Literature Review," ACM Computing Surveys, vol. 52, no. 6, pp. 1–42, 2019. [Online]. Available: https://dl.acm.org/doi/abs/10.1145/3345317

[2] H. A. Chowdhury and D. K. Bhattacharyya, "Plagiarism: Taxonomy, Tools and Detection Techniques," arXiv preprint, arXiv:1801.06323, 2018. [Online]. Available: https://arxiv.org/abs/1801.06323

[3] S. V. Moravvej, S. J. Mousavirad, D. Oliva, and F. Mohammadi, "A Novel Plagiarism Detection Approach Combining BERT-based Word Embedding, Attention-based LSTMs and an Improved Differential Evolution Algorithm," arXiv preprint, arXiv:2305.02374, 2023. [Online]. Available: https://arxiv.org/abs/2305.02374

[4] O. Kamat, T. Ghosh, J. Kalaivani, V. Angayarkanni, and P. Rama, "Plagiarism Detection Using Machine Learning," arXiv preprint, arXiv:2412.06241, 2024. [Online]. Available: https://arxiv.org/abs/2412.06241

[5] C.-Y. Chen, J.-Y. Yeh, and H.-R. Ke, "Plagiarism Detection using ROUGE and WordNet," arXiv preprint, arXiv:1003.4065, 2010. [Online]. Available: https://arxiv.org/abs/1003.4065

[6] K. S. Sonawane and S. Prabhudeva, "Plagiarism Detection Using Machine Learning," International Research Journal of Modernization in Engineering, Technology and Science, vol. 5, no. 5, 2023. [Online]. Available: https://www.irjmets.com/uploadedfiles/paper/issue_5_may_2023/39442/final/fin_irjmets1684466327.pdf

[7] O. E. Taylor, "Plagiarism Detection System using an Enhanced String Matching Algorithm," Journal of Scientific and Engineering Research, vol. 8, no. 4, pp. 99–105, 2021. [Online]. Available: https://www.researchgate.net/publication/362062474

[8] A. Ahnaf, H. M. M. Hasan, N. S. Sworna, and N. Hossain, "An Improved Extrinsic Monolingual Plagiarism Detection Approach of the Bengali Text," 2023. [Online]. Available: https://www.researchgate.net/publication/369673129

[9] A. Băutu and E. Băutu, "PlagZap: A Textual Plagiarism Detection System for Student Assignments Built with Open-Source Software," in Proc. Int. Conf. on e-Learning, 2019, pp. 123–130. [Online]. Available: https://www.researchgate.net/publication/325628538

[10] H. Ramnial, S. Panchoo, and S. Pudaruth, "Authorship Attribution Using Stylometry and Machine Learning Techniques," in Proc. Int. Conf. on Computer, Communication and Control Technology, 2016, pp. 1–6. [Online]. Available: https://www.researchgate.net/publication/283862723