International Journal of Multidisciplinary Engineering in Current Research - IJMEC
Volume 11, Issue 7, July-2026, http://ijmec.com/, ISSN: 2456-4265

Plagiarism Analyzer Using Python

Ms. Kazi Nikhat Parvin M[1], P. Divya[2], Pallavi[3], Madiha Iffath[4]
[1]Associate Professor; Department of Electronics and Communication Engineering, Bhoj Reddy Engineering College for Women Hyderabad, India
[2,3,4]B. Tech Students: Department of Electronics and Communication Engineering, Bhoj Reddy Engineering College for Women Hyderabad, India
Mail Id: divyapathuri513@gmail.com[2],madihaiffath19@gmail.com[4],pallavibangale.22@gmail.com[3]
Accepted 4 July 2026

Abstract
Plagiarism detection has become an essential requirement in academic and professional environments to ensure originality and integrity of content. This project presents the design and development of a plagiarism analyzer using Python, which compares textual data to identify similarities between documents. The system utilizes Natural Language Processing (NLP) techniques such as tokenization, stop-word removal, and text normalization to preprocess the input data.
The core functionality is based on similarity measurement algorithms like cosine similarity and sequence matching, which evaluate the degree of resemblance between two or more text files. Python libraries such as NLTK, scikit-learn, and difflib are used to efficiently process and analyze the text. The system outputs a similarity percentage that indicates the level of plagiarism present in the documents.
This plagiarism analyzer is simple, efficient, and scalable, making it suitable for applications in educational institutions, content verification systems, and research work. It helps users detect copied content quickly and promotes the creation of original material.
Plagiarism refers to copying someone else's work without proper acknowledgment. With the growth of digital content, detecting plagiarism has become very important in academics, research, and content creation. A plagiarism analyzer helps in identifying copied or similar content between documents.
This project uses Python to build a simple and efficient plagiarism detection system that compares text files and calculates similarity scores.
The system utilizes Natural Language Processing (NLP) techniques to preprocess the input text, including tokenization, stop-word removal, stemming, and normalization. These steps help in improving the accuracy of comparison by eliminating irrelevant and redundant data. The processed text is then transformed into numerical vectors using techniques such as Term.

Keywords: *Plagiarism Detection, Natural Language Processing (NLP), Python, Text Similarity, Cosine Similarity, TF-IDF, Tokenization, NLTK, Scikit-learn, Sequence Matching, Text Mining, Document Analysis.*

Introduction

In today's digital world, a vast amount of information is easily accessible through the internet. While this makes learning and research easier, it also increases the chances of plagiarism, when individuals copy content from sources without giving proper c reduces the originality of work.
The Plagiarism Analyzer using Python is a software application developed to automatically detects similarities between text documents.
In this project, methods such as Term Frequency–Inverse Document Frequency (TF-IDF) and Cosine Similarity are used to calculate similarity scores between documents. The program reads multiple text files, processes the content, and then displays the percentage of similarity between them. High similarity scores may indicate possible plagiarism.
The main aim of this project is to build a simple and efficient plagiarism detection tool using Python that can help teachers, students, and researchers verify the originality of documents. By automating the comparison process, the system saves time, improves accuracy, and helps maintain academic integrity. In the modern digital era, the availability of vast information on the internet has made it easier for people to access knowledge for education, research, and content creation. However, this accessibility has also increased the risk of plagiarism, which refers to the act of copying someone else's work, ideas or text without giving proper credit. Plagiarism is considered unethical and can lead to serious consequences in academic, professional, and research environments. Therefore, it is important to ensure that documents and assignments are original and free from copied content. Detecting plagiarism manually can be a challenging and time-consuming task, especially when dealing with large amounts of text or multiple documents. Traditional methods of checking documents require careful reading and comparison, which is not always efficient. To overcome these challenges, automated systems can be developed to detect similarities between documents quickly and accurately. The software tool designed to identify similarities between text documents by using Natural Language Processing (NLP).

Literature Survey

Plagiarism detection has become an important area of research due to the rapid growth of digital content, online learning platforms, and academic publications. Ensuring the originality of written material is essential in educational institutions, research organizations, and professional environments. Over the years, researchers have proposed various plagiarism detection techniques ranging from simple string matching and document fingerprinting methods to advanced Natural Language Processing (NLP) and machine learning approaches. Traditional methods mainly focus on identifying exact text matches, whereas modern approaches analyze semantic similarity and contextual relationships to improve detection accuracy. Similarity measurement techniques such as **Cosine Similarity**, **Jaccard Similarity**, **Sequence Matching**, and the **Winnowing Algorithm** have been widely adopted for comparing textual documents. These techniques enable the identification of both exact and partial similarities, making plagiarism detection systems more effective and reliable.

The proposed plagiarism analyzer is developed using a lightweight and efficient technology stack that combines Python with open-source libraries for text processing and similarity analysis. **Python** is selected as the primary programming language because of its simple syntax, extensive library support, cross-platform compatibility, and strong community support. It is particularly suitable for Natural Language Processing applications due to the availability of powerful libraries such as **NLTK**, **scikit-learn**, and **difflib**. These libraries provide built-in functionalities for text preprocessing, including tokenization, stop-word removal, stemming, and normalization, as well as feature extraction and similarity computation. The use of Python enables rapid development, simplified debugging, and efficient implementation of plagiarism detection algorithms. Overall, the combination of NLP techniques and Python-based tools provides a scalable, accurate, and cost-effective solution for detecting textual similarities and promoting originality in academic and professional documents.

Problem Statement

In the digital age, the easy availability of online information has led to a significant increase in the misuse of content through copying and unauthorized reuse. Plagiarism has become a major concern in academic institutions, research communities, and content creation industries. Ensuring originality in documents is essential to maintain ethical standards

Traditional methods of detecting plagiarism, such as manual comparison of documents, are time-consuming, inefficient, and prone to human error. Existing plagiarism detection tools are often expensive, require internet access, or depend on large proprietary databases, making them inaccessible to many students and small organizations.

Moreover, detecting similarity between documents is not a straightforward task, as plagiarism can occur in various forms, including direct copying, partial modification, and paraphrasing. This creates a need for an automated system that can efficiently analyze and compare textual data to identify similarities

The system should be simple to use, provide accurate results for textual similarity, and serve as a practical solution for academic and small-scale applications.

Traditional methods of detecting plagiarism, such as manual comparison of documents, are time-consuming, inefficient, and prone to human error. Existing plagiarism detection tools are often expensive, require internet access, or depend on large proprietary databases, making them inaccessible to many students and small organizations.

Moreover, detecting similarity between documents is not a straightforward task, as plagiarism can occur in various forms, including direct copying, partial modification, and paraphrasing. This creates a need for an automated system that can efficiently analyze and compare textual data to identify similarities

The system should be simple to use, provide accurate results for textual similarity, and serve as a practical solution for academic and small-scale applications.

With the rapid expansion of digital technologies and internet access, vast amounts of textual information are readily available. While this has improved learning and research, it has also made it easier to copy and reuse content without proper acknowledgment. This has led to a rise in plagiarism across academic, professional, and online platforms. Plagiarism undermines originality, intellectual property rights, and academic integrity. Detecting plagiarism manually is not feasible when dealing with large volumes of documents. There is aclear need for an automated system that can efficiently compare documents and identify similarities.

the increasing prevalence of plagiarism and the limitations of existing detection methods highlight the need for a simple, efficient, and accessible plagiarism detection system. The proposed Python-based solution aims to address these issues by providing an automated and reliable method for identifying similarities in text documents.

Traditional methods of detecting plagiarism, such as manual comparison of documents, are time-consuming, inefficient, and prone to human error. Existing plagiarism detection tools are often expensive, require internet access, or depend on large proprietary databases, making them inaccessible to many students and small organizations.

Moreover, detecting similarity between documents is not a straightforward task, as plagiarism can occur in various forms, including direct copying, partial modification, and paraphrasing. This creates a need for an automated system that can efficiently analyze and compare textual data to identify similarities

In the digital age, the easy availability of online information has led to a significant increase in the misuse of content through copying and unauthorized reuse. Plagiarism has become a major concern in academic institutions, research communities, and content creation industries. Ensuring originality in documents is essential to maintain ethical standards

The next challenge is measuring similarity between documents. This is addressed using TF-IDF (Term Frequency–Inverse Document Frequency) and cosine similarity, which convert textual data into numerical vectors and calculate the similarity score. Libraries such as scikit-learn and NLTK are used to implement these techniques efficiently.

Another problem is handling multiple documents and generating meaningful results. The system solves this by comparing documents in pairs and producing similarity percentages that clearly indicate the level of plagiarism. This makes the output easy to understand and useful for users such NumPy supports efficient numerical computations and array operations, which are required during similarity calculations.

Design of Plagiarism Analyzer Using Python

The block diagram of the Plagiarism Analyzer using Python provides a visual representation of the overall system architecture and workflow. It illustrates how different components of the system as students and educators.

Software Requirements

Python is used as the primary programming language for this project due to its simplicity, readability,and extensive support for text processing. It provides a wide range of libraries that simplify theimplementation of Natural Language Processing (NLP) techniques and similarity algorithms. Pythonalso supports rapid development and easy debugging, making it suitable for building such analyticalsystems.

NLTK (Natural Language Toolkit):

This library is used for preprocessing textual data. It helps in breaking down text into smaller components (tokenization), removing common words (stop words), and normalizing the text, which

improves the accuracy of similarity detection. Scikit-learn:

Scikit-learn is used for transforming text into numerical representations using techniques such as TF-IDF (Term Frequency-Inverse Document Frequency). It also provides tools to calculate similaritymeasures like cosine similarity, which is essential for comparing documents.

NumPy:

interactwith each other to perform plagiarism detection effectively.

The system begins with user input, where the text or document to be analyzed is provided. This input isthen passed to the preprocessing module, which performs operations such as tokenization, removal of stop words, and text normalization.

Block Diagram

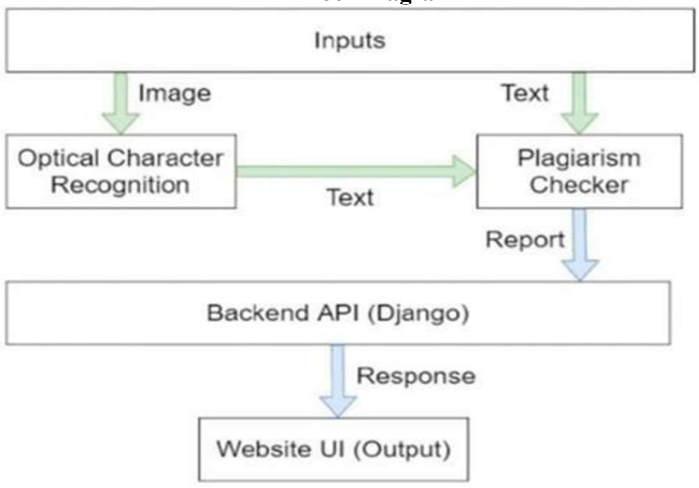

Figure 1; Block Diagram

Working Methodology

Input Module

The system accepts input as typed/uploaded text documents and images containing text such as scanned files or photos.

This flexibility enables plagiarism analysis for both

digital text and image-based content.

Optical Character Recognition (OCR) Module

If the input is an image, the OCR module converts scanned or photographed text into machine-readable format.

This conversion allows image-based documents to be processed for plagiarism detection.

Plagiarism Checker Module

This core module preprocesses text using NLP techniques like tokenization and stop-word removal.

The processed text is converted into numerical vectors using TF-IDF for similarity measurement and plagiarism detection.

Backend API (Django)

The backend built with Django Software Foundation framework manages communication between the plagiarism engine and user interface.

It processes user requests, performs analysis, and returns plagiarism results.

Website User Interface (Output)

The website interface displays plagiarism percentage and similarity reports as final output. Users can view highlighted sections showing similar documents or text segments.

Results

The developed plagiarism analyzer successfully compares textual documents and presents the results in the form of similarity percentages, which indicate the extent of plagiarism between the given documents. A higher similarity percentage represents a greater degree of copied content, whereas a lower percentage indicates that the documents contain more original text. The system effectively detects both exact matches and partial similarities, making it suitable for basic plagiarism detection in academic, research, and content verification applications.

The analyzer calculates similarity scores using **Term Frequency–Inverse Document Frequency (TF-IDF)** and **Cosine Similarity**, enabling accurate comparison of textual data by considering both the frequency and importance of words within the documents. Based on the computed similarity score, the system classifies the results into different levels, such as **high**, **moderate**, or **low similarity**, allowing users to interpret the plagiarism level easily. In addition to providing an overall similarity percentage, the system identifies common words and phrases shared between documents while also highlighting unique content, helping users distinguish between duplicated and original text.

The proposed system demonstrates efficient performance by processing input documents quickly and generating similarity results in real time. It supports the comparison of multiple documents and produces similarity scores for each document pair, making it suitable for educational institutions and small-scale document verification systems. The user-friendly output format enables users to understand the results without requiring technical knowledge.

Experimental evaluation shows that the analyzer consistently produces reliable similarity scores for identical input documents, ensuring repeatability and dependable performance. The use of Python libraries such as **NLTK**, **scikit-learn**, and **difflib** contributes to accurate text preprocessing and similarity analysis. Although the system performs well in detecting exact copies and partial text matches, it has limited capability in identifying heavily paraphrased content or understanding semantic relationships between sentences. Despite these limitations, the proposed plagiarism analyzer provides an efficient, accurate, and cost-effective solution for automated plagiarism detection and promotes originality in academic and professional writing.

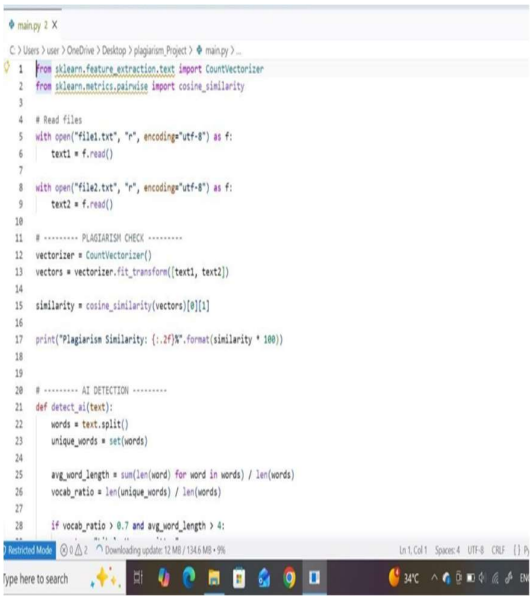

Figure 2; implementation of text similarity and AI detection

The above figure shows the implementation of a plagiarism detection and AI-based text classification system developed using Python in Visual Studio Code. Initially, the system reads twoinput text files (file1.txt and file2.txt) using file handling techniques. These texts are then processed using the CountVectorizer, which converts the textual data into numerical vectors based on word frequency.

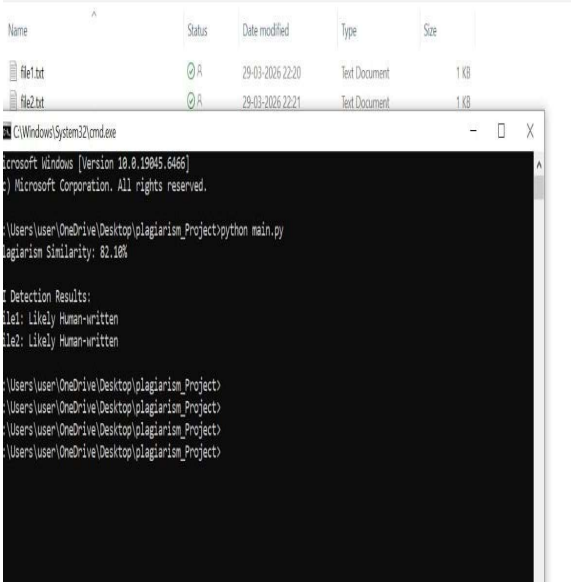

Figure: 3; Command Prompt Display of Similarity and AI Analysis Results

The output shows that the plagiarism similarity is 82.10%, which indicates a very high level of similarity between the two files. Generally, a similarity score above 70% suggests that the content is either directly copied or heavily paraphrased. Therefore, it can be concluded that both files share a significant portion of common content.

In addition to plagiarism detection, the program also performed AI detection. The results indicate that both File 1 and File 2 are likely human-written

Discussion

The plagiarism analyzer developed using Python is an effective tool for detecting similarities between textual documents. It is based on techniques from Natural Language Processing, which help in analyzing and processing human language data. The system works by taking input documents and preprocessing the text through steps such as converting all characters to lowercase, removing punctuation, and eliminating common stopwords. This ensures that only meaningful words are considered during comparison.

Conclusion

The Plagiarism Analyzer developed using Python successfully demonstrates the ability to compare textual data and identify similarity between documents. By utilizing Natural Language Processing techniques such as CountVectorizer and cosine similarity, the system efficiently converts text into numerical form and measures the degree of similarity.

The results show that the system can effectively detect high levels of plagiarism and provide meaningful insights into content overlap. In addition, the integration of a basic AI detection mechanism adds value by attempting to classify whether the text is human-written or machine-generated.

Overall, the project highlights the practical use of Python in building a plagiarism detection tool and serves as a foundation for developing more advanced and reliable text analysis systems.

However, the system has certain limitations, particularly in identifying paraphrased content and understanding deep semantic meaning. Despite these challenges, the project serves as a strong foundation for further improvements. Future enhancements can include integrating advanced models such as BERT, supporting multiple file formats, and improving overall accuracy and performance.

This project highlights the effectiveness of Python as a programming language for developing text analysis tools, especially with the support of powerful libraries such as scikit-learn and NLTK. The system is capable of identifying exact matches as well as partial similarities, making it suitable for academic use, content verification, and basic plagiarism detection tasks. Additionally, the integration of AI-based detection provides further insight into whether the content is likely human-written, adding another layer of analysis to the system.

Despite its advantages, the analyzer has certain limitations. It is less effective in detecting advanced forms of plagiarism such as paraphrasing, synonym replacement, and structural modifications of sentences. The lack of deep semantic understanding restricts its ability to interpret context and meaning at a higher level.

In conclusion, the plagiarism analyzer is a valuable tool that highlights the importance of originality in content creation while showcasing the practical applications of Python in text analysis and document comparison.

The analyzer is effective in detecting exact matches and partially copied content, making it useful for academic and content verification purposes.

Future Scope

The future scope of this project includes several potential enhancements and extensions:

The plagiarism analyzer developed using Python has significant potential for further improvement and expansion. While the current system effectively detects direct and partial similarities using techniques from Natural Language Processing, future enhancements can make it more advanced, accurate, and scalable.

One major area of improvement is the integration of advanced language models such as BERT, which can understand the semantic meaning of text. This would allow the system to detect paraphrased content and context-based similarities more effectively, overcoming one of the key limitations of

the current model.

The system can also be extended to support multiple file formats such as PDF, DOCX, and web content, making it more versatile and user-friendly. In addition, developing a graphical user interface (GUI) or web-based application would improve accessibility, allowing users to upload and compare documents easily without requiring programming knowledge.

Another important future enhancement is improving performance and scalability. By optimizing algorithms and using cloud-based technologies, the analyzer can handle large datasets and multiple document comparisons efficiently. Integration with databases can also help in storing and managing large volumes of documents for continuous plagiarism checking.

The future scope of a plagiarism analyzer using Python is highly promising due to advancements in artificial intelligence and natural language processing. Modern systems can evolve from simple text matching to semantic analysis, enabling detection of paraphrased and idea-based plagiarism.

References

[1] K. Kalebu, "Plagiarism Checker Python," *GitHub Repository*, 2023.

[2] Scriptographers, "UnPlag: Plagiarism Detection System," *GitHub Repository*, 2021.

[3] "Detection of Document Plagiarism Using Winnowing Algorithm and K-Gram Method," *International Journal of Advanced Computer Science*, 2017.

[4] "Detection of Text Similarity for Plagiarism Using Winnowing Algorithm and Jaccard Coefficient," *International Journal of Computer Applications*, vol. 176, no. 39, pp. 1–6, 2020.

[5] M. Schleimer, D. S. Wilkerson, and A. Aiken, "Winnowing: Local Algorithms for Document Fingerprinting," in *Proceedings of the 2003 ACM SIGMOD International Conference on Management of Data*, San Diego, CA, USA, 2003, pp. 76–85.

[6] "Plagiarism Detection Using Python," *GeeksforGeeks*, 2025.

[7] "Simple Plagiarism Detector in Python Using Difflib," *GeeksforGeeks*, 2025.