ASSIGNMENT PLAGIARISM CHECKER PROJECT

UNIVERSITY OF ENGINEERING & MANAGEMENT, JAIPUR

Submitted in the partial fulfillment of the degree of

BACHELOR OF TECHNOLOGY

In

COMPUTER SCIENCE & ENGINEERING

Under

UNIVERSITY OF ENGINEERING & MANAGEMENT, JAIPUR

BY

Arya Kumar Johary

University Roll no: 12021002026028

Registration Number: 204202100200196

&

Abhay Prasad

University Roll no: 12021002026024

Registration Number: 204202100200192

UNDER THE GUIDANCE OF

Asst. Prof. Sagarika Ghosh & Asst. Prof. Hriday Banerjee

COMPUTER SCIENCE & ENGINEERING

UNIVERSITY OF ENGINEERING & MANAGEMENT, JAIPUR

Approval Certificate

This is to certify that the project report entitled "**Plagiarism Checker**" submitted by Abhay Prasad (12021002026024) & Arya Kumar Johary (12021002026028) in partial fulfillment of the requirements of the degree of **Bachelor of Technology** in **Computer Science & Engineering** from **University of Engineering and Management, Jaipur** was carried out in a systematic and procedural manner to the best of our knowledge. It is a bona fide work of the candidate and was carried out under our supervision and guidance during the academic session of 2021-2025.

_____

Asst. Prof. Sagarika Ghosh
Project Guide, Assistant Professor (CSE)
UEM, JAIPUR

_____

Asst. Prof. Hriday Banerjee
Project Guide, Assistant Professor (CSE)
UEM, JAIPUR

_____

Prof. Mrinal Kanti Sarkar
HOD (CSE)
UEM, JAIPUR

_____

Prof. A. Mukherjee
Dean
UEM, JAIPUR

ACKNOWLEDGEMENT

The endless thanks goes to Lord Almighty for all the blessings he has showered onto us, which has enabled us to write this last note in our research work. During the period of our research, as in the rest of our life, we have been blessed by Almighty with some extraordinary people who have spun a web of support around us. Words can never be enough in expressing how grateful we are to those incredible people in our life who made this thesis possible. We would like an attempt to thank them for making our time during my research in the Institute a period we will treasure. We are deeply indebted to our research supervisors, Prof. Sagarika Ghosh and Prof. Hriday Banerjee, who gave us such an interesting thesis topic. Each meeting with them added in valuable aspects to the implementation and broadened our perspective. They have guided us with his invaluable suggestions, lightened up the way in our darkest times and encouraged us a lot in the academic life.

Arya Kumar Johary & Abhay Prasad

ABSTRACT

The proposed Plagiarism Checker utilizes compares textual content and identify similarities, ensuring a comprehensive analysis of academic documents. Key features of the Plagiarism Checker include a user-friendly interface, scalability to handle large datasets, and real-time feedback on the level of plagiarism detected. The system employs advanced linguistic analysis to identify paraphrased content and employs a robust algorithm that can adapt to different writing styles and languages.

The Plagiarism Checker is implemented with a focus on efficiency, accuracy, and user accessibility. In conclusion, the Plagiarism Checker project provides a comprehensive and sophisticated solution to address the growing concerns of plagiarism in academic settings. By combining cutting-edge technologies and robust algorithms, this tool aims to promote academic integrity, uphold ethical standards, and foster a culture of originality and intellectual honesty within educational institutions.

Table of Contents

List of Figure

1. CHAPTER

INTRODUCTION

Plagiarism is a serious academic offense that involves using someone else's words or ideas without proper acknowledgment. It can have negative consequences for both students and teachers, such as loss of credibility, lower grades, or even legal actions. Therefore, it is important to have effective tools and methods to detect and prevent plagiarism in written work. In this project, we will present a plagiarism checker program that uses a combination of TF-IDF vectorizer and cosine similarity to compare text documents and identify potential plagiarism.

TF-IDF vectorizer is a technique that converts a collection of text documents into a matrix of numerical values that represent the importance of each word in each document. This way, it can capture the uniqueness and relevance of each word in each document.

Cosine similarity is a measure that calculates the angle between two vectors, in this case, the TF-IDF vectors of two documents. It ranges from 0 to 1, where 0 means the vectors are orthogonal (no similarity) and 1 means the vectors are identical (maximum similarity). By comparing the cosine similarity of different pairs of documents, we can determine how similar they are in terms of their word usage and content.

By using these techniques, we can create a plagiarism checker program that can help us to ensure originality and integrity in writing. The program can take one or more text documents as input, and output a similarity score for each pair of documents. The program can be useful for students, teachers, researchers, and writers who want to check their work for plagiarism and avoid academic dishonesty.

2. CHAPTER

LITERATURE REVIEW

Plagiarism undermines the credibility and integrity of scholarly work. The development and implementation of plagiarism checker systems have become essential to prevent and detect plagiarism in academic writing. This literature review examines existing research and developments in the field of plagiarism checker systems, with a particular focus on projects utilizing Python as a primary programming language.

A study in 2014 done by Martins, Vítor T., Fonte, Daniela, Henriques, Pedro Rangel Cruz and Daniela da suggests following methodologies for plagiarism detection:

Attribute-based Methodology: This method involves calculating metrics from the source code, such as the size of the code in terms of characters, words, and lines. For instance, if two source codes have significantly different sizes, they can be identified using this attribute-based approach.

Token-based Methodology: In this approach, the source code is tokenized (broken down into meaningful units) and hashed. The resulting hashes are then used to create fingerprints for comparison. This method is efficient and was employed by researchers.

Structure-based Methodology: Here, the source code is transformed into an Internal Intermediate Representation (IR), like an Abstract Syntax Tree (AST) or Program Dependence Graph (PDG). This representation is then used for accurate comparisons, enabling a detailed analysis of the code's structure.

Further, the study by Hakkun Elmunsyah, Hary Suswanto, Khoirudin Asfani and Wahyu Hidayat conducted in 2018 addresses the negative impact of information technology on education, focusing

on soft skills like honesty and self-confidence, and copyright awareness. Using the ADDIE model, it evaluates a Plagiarism Checker software's effectiveness in combating plagiarism among Vocational High School students in Malang. Results show high feasibility (above 85%) in detecting similarity in scientific papers.

While the development of plagiarism checker systems using Python shows potential, challenges such as scalability, performance, and ethical issues remain. Research by Angelos Rodafinos in 2018 discusses the emerging trends and challenges in the field of plagiarism checker systems and suggests directions for future research, including the adoption of cloud computing and big data analytics for improved scalability and performance.

2.1 Vectors Numerizing Array

In the context of the Plagiarism Checker project, the process of numerizing arrays involves the representation of textual content in a numerical form through the use of vectors. This technique is crucial for comparing and analyzing documents to identify similarities and potential instances of plagiarism.

- **Vectorization Process**

The algorithm employed in this project utilizes the Term Frequency-Inverse Document Frequency (TF-IDF) vectorization technique. The following steps are involved in the vectors numerizing array process:

- **Text Preprocessing**

Raw text data from each document is preprocessed to remove stop words, punctuation, and irrelevant characters.

Tokenization is applied to break down the text into individual words or phrases.

- **TF-IDF Vectorization**

The TF-IDF vectorizer is applied to convert the tokenized text into numerical vectors. TF-IDF assigns weights to each term based on its frequency in a document relative to its frequency across all documents.

- **Creation of Numerical Arrays**

The resulting TF-IDF vectors form numerical arrays representing the content of each document.

These numerical arrays serve as a basis for calculating the similarity between documents.

- **Similarity Calculation**

The project utilizes cosine similarity to measure the similarity between pairs of documents. Cosine similarity is a metric that calculates the cosine of the angle between two vectors, providing a measure of similarity between documents.

By numerizing arrays through vectorization and employing cosine similarity, the Plagiarism Checker project can effectively identify patterns and similarities in textual content, contributing to the accurate detection of potential plagiarism.

2.2 Methodology To Find Similarity

The project utilizes cosine similarity to measure the similarity between pairs of documents. Cosine similarity is a metric that calculates the cosine of the angle between two vectors, providing a measure of similarity between documents.

By numerizing arrays through vectorization and employing cosine similarity, the Plagiarism Checker project can effectively identify patterns and similarities in textual content, contributing to the accurate detection of potential plagiarism.

2.3 Why Cosine Similarity?

Cosine similarity is a widely used metric for comparing the similarity between two vectors in various applications, including natural language processing and document analysis. Here are some reasons why cosine similarity is often chosen for tasks like plagiarism detection:

- **Scale Invariance**

Cosine similarity is scale-invariant, meaning it measures the cosine of the angle between two vectors rather than their magnitudes. This property makes it robust when dealing with documents of varying lengths, as it focuses on the direction of the vectors rather than their absolute values.

- **Angle Measure**

Cosine similarity calculates the cosine of the angle between two vectors, providing a measure of orientation rather than distance. A similarity score of 1 indicates that the vectors are perfectly aligned, while a score of 0 means they are orthogonal (no similarity).

- **Efficiency**

Computationally, cosine similarity is efficient and relatively straightforward to implement. It involves simple vector operations, making it a practical choice for large datasets and real-time applications.

- **Handling Sparse Data**

In document analysis, textual data is often represented as sparse vectors where most elements are zero. Cosine similarity is well-suited for sparse data because it considers only the non-zero elements, ignoring the zero entries in the vectors.

- **Common Term Handling**

Cosine similarity naturally handles common terms that might appear in multiple documents. TF-IDF (Term Frequency-Inverse Document Frequency) weighting, often used in conjunction with cosine similarity, helps de-emphasize common terms while emphasizing rare terms.

- **Interpretability**

The cosine similarity score ranges from -1 to 1, where 1 indicates perfect similarity, 0 indicates no similarity, and -1 indicates perfect dissimilarity. This range is intuitive and facilitates easy interpretation of results.

- **Widely Adopted**

Cosine similarity has been successfully employed in various natural language processing and information retrieval tasks. Its proven effectiveness makes it a standard choice for measuring document similarity.

2.4 Using tkinter for GUI

- **User-Friendly Interface**

Ensure that the interface is intuitive and easy to navigate. Include clear labels, concise instructions, and a logical flow for users to browse files, initiate plagiarism checks, and view results.

- **Visual Feedback for Similarity**

Use color-coded or visually distinguishable elements to represent the level of similarity between documents. For example, you could use different colors or shading to indicate low, moderate, and high levels of similarity.

- **File Drag-and-Drop Feature**

Allow users to drag and drop files directly onto the application for added convenience. This can streamline the process of selecting files for plagiarism checks.

- **Real-Time Analysis**

Provide real-time feedback as the plagiarism analysis progresses. Users may appreciate immediate results for smaller document sets, making the tool more interactive.

- **File Information Display**

Display relevant information about selected files, such as file names, sizes, and the number of words or characters. This adds transparency and helps users verify their selections.

- **Error Handling and Notifications**

Implement clear error messages and notifications to inform users about issues such as invalid file formats, unsuccessful uploads, or any other potential errors.

3. CHAPTER

METHODOLOGY

3.1 Vectorizing the String Data

The TF-IDF (Term Frequency-Inverse Document Frequency) vectorizer is a text processing technique that converts a collection of documents into a numerical format suitable for machine learning. Here's a brief overview.

TF-IDF vectorization involves the following steps:

1. **Tokenization:** The documents are broken down into individual words or tokens.

2. **Term Frequency (TF)**: Measures how frequently a term appears in a document.

3. **Document Frequency (DF)**: Counts how many documents contain a specific term.

4. **Inverse Document Frequency (IDF**): Weights terms based on their rarity across the entire document collection.

5.**TF-IDF Calculation**: Combines TF and IDF to assign a score to each term in each document.

6. **Normalization:** Optionally, the resulting matrix can be normalized for consistent scaling.

The TF-IDF matrix represents documents as vectors in a high-dimensional space, facilitating text-based comparisons in machine learning applications.

3.2 Finding Cosine Similarity

In scikit-learn (sklearn), the implementation of cosine similarity is available through the cosine_similarity function in the “metrics. pairwise” module. This function efficiently computes the cosine similarity between pairs of vectors or between rows in a matrix. The input can be dense or sparse matrices.

Here's a concise explanation:

The cosine_similarity function in scikit-learn uses the dot product of vectors and their L2 norms to calculate the cosine similarity. Given two vectors u and v, the cosine similarity cos(Ɵ) is computed as the dot product of u and v divided by the product of their L2 norms:

$\cos(Ɵ)=u.v / |u|.|v|$

The result is a similarity matrix where each element represents the cosine similarity between corresponding pairs of vectors. This implementation is efficient and well-suited for various applications, such as text analysis and clustering, where measuring similarity between data points

3.3 Creation Of GUI

The provided code creates a Tkinter-based graphical user interface (GUI) for a Plagiarism Checker project. Key functions include:

Browse_Files: Enables users to select multiple text files for plagiarism analysis. Selected file paths are displayed in a Listbox.

Check_Selected_Files: Initiates the plagiarism check for the selected files, displaying results in the Text widget. It checks for a minimum of two selected files.

Clear_List: Clears the Listbox and result Text widget to reset the interface for a new analysis.

Disable_Boxes And Enable_Boxes: Control the state (disabled or normal) of the Listbox and Text widget, ensuring user interaction during different phases of the analysis.

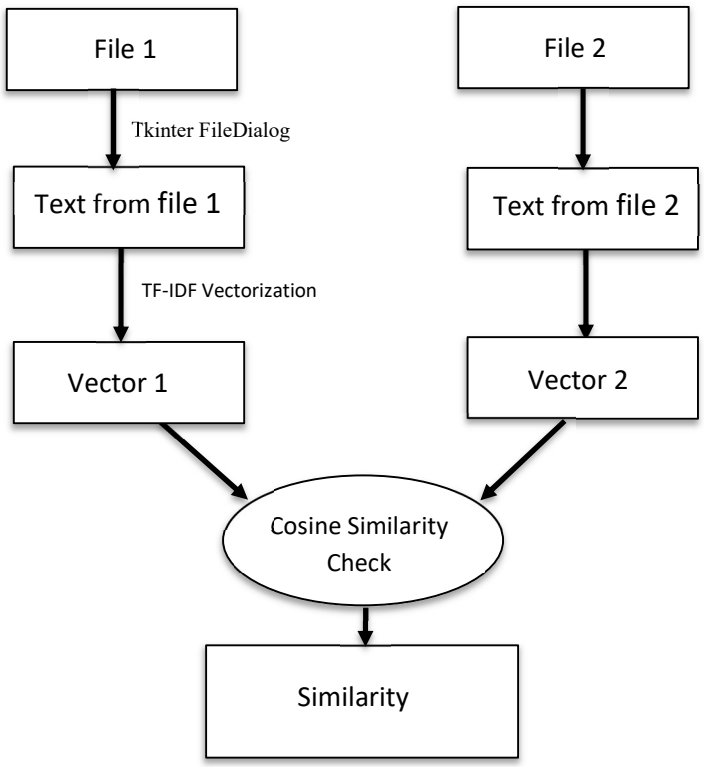

Figure 3.4.1 – Proposed Model for the project

4. CHAPTER

RESULTS & DISCUSSION

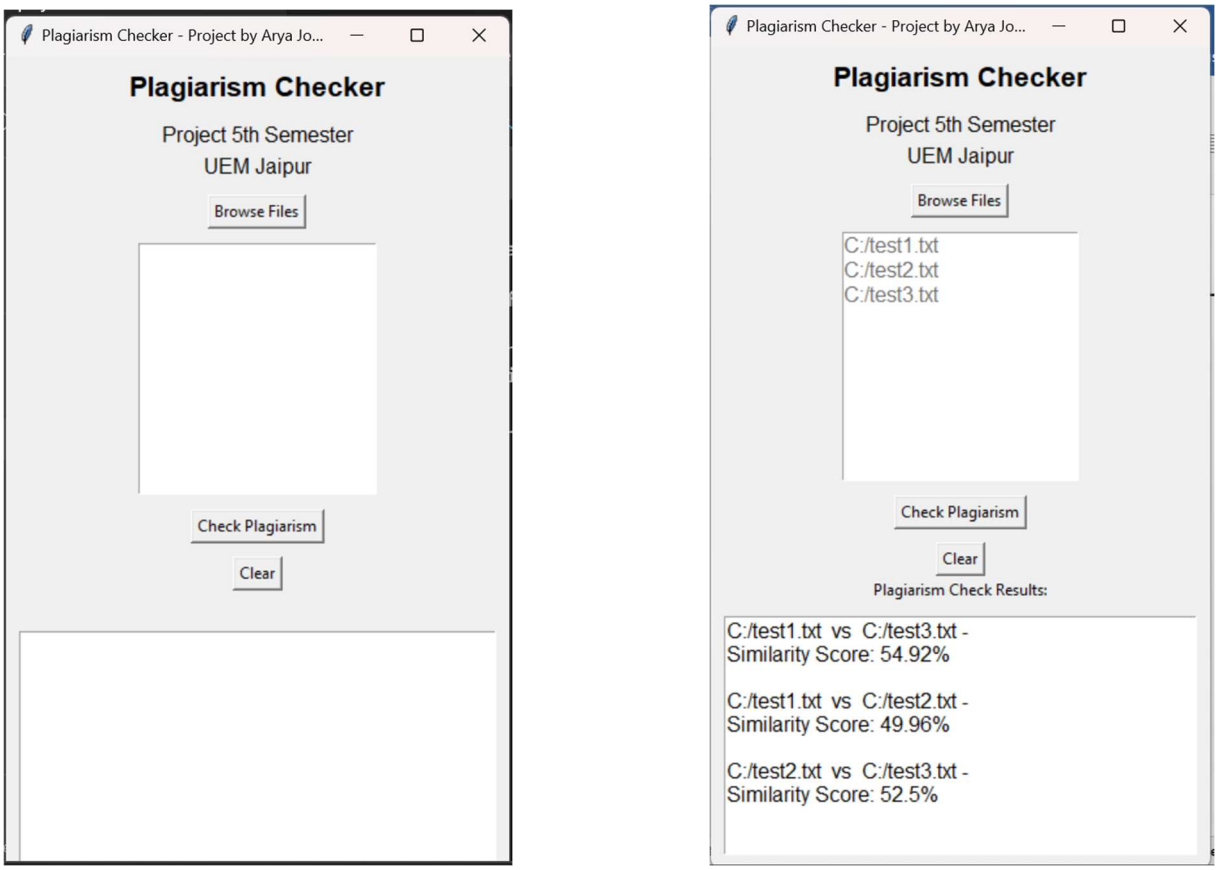

Fig. 4.1 - GUI for Plagiarism Checker

CONCLUSION AND FUTURE SCOPE

The Plagiarism Checker project is a Python program that detects similarities among text documents. It uses cosine similarity to measure the likeness between documents and shows the results in a user-friendly interface. Users can browse, select, and analyze multiple files for potential plagiarism. The project supports academic integrity by encouraging originality and ethical writing. The Plagiarism Checker project is a useful tool for education and writing.

Some possible future scope for this project are:

- The project can be extended to detect plagiarism in other types of content, such as images, audio, video, or code, by using appropriate similarity metrics and techniques.
- The project can be improved to handle different languages, formats, and sources of text documents, by using natural language processing, text mining, and web scraping tools.
- The project can be integrated with online learning platforms, academic journals, or other content providers, to provide a convenient and reliable service for checking plagiarism and ensuring originality.
- The project can be enhanced to provide feedback and suggestions for improving the quality and creativity of the text documents, by using natural language generation, sentiment analysis, and text summarization tools.

BIBLIOGRAPHY

1. Tkinter Documentation. (https://docs.python.org/3/library/tkinter.html)

2. Python Documentation. (https://docs.python.org/3/)

3. Cosine Similarity Documentation.

   (https://scikit-learn.org/stable/modules/generated/sklearn.metrics.pairwise.cosine_similarity.html)

4. TF – IDF Vectorizer Documentation.

   (https://scikit-

   learn.org/stable/modules/generated/sklearn.feature_extraction.text.TfidfVectorizer.html)

5. The Effectiveness of Plagiarism Checker Implementation in Scientific Writing for Vocational

   High School

   (https://www.atlantis-press.com/proceedings/aptekindo-18/25903493)

6. Plagiarism Detection: A Tool Survey and Comparison

   (https://repositorium.uminho.pt/bitstream/1822/64358/1/14.pdf)

7. Plagiarism Management: Challenges, Procedure and Workflow Automation

   (https://www.ijello.org/Volume14/IJELLv14p159-175Rodafinos5036.pdf)