IJCRT.ORG ISSN : 2320-2882

INTERNATIONAL JOURNAL OF CREATIVE RESEARCH THOUGHTS (IJCRT)

An International Open Access, Peer-reviewed, Refereed Journal

TEXTGUARDIANSHIP

PYTHON & AI-POWERED PLAGIARISM DETECTION

[1]Milind Patekar, [2]Chinmay Pathak, [3]Sushant Patil, [4]Devendra Shete, [5]Neelam Jain

[1]Final Year BE Student, [2]Final Year BE Student, [3]Final Year BE Student, [4]Final Year BE Student, [5]Assistant Professor

[1]Department of Artificial Intelligence and Data Science,

[1]Ajeenkya DY Patil School of Engineering, Pune, India

Abstract: In the age of the internet, maintaining the authenticity of text content has become more and more important for academic institutions, content developers, and professionals. This paper introduces TextGuardianship, a Python and AI-based plagiarism detection tool that aims to provide a sound and effective means of detecting potential plagiarism. The system utilizes cutting-edge Natural Language Processing (NLP) methods and machine learning models, such as TF-IDF vectorization, cosine similarity, and semantic analysis, to scan and compare text data with high precision. Main aspects of TextGuardianship are the provision for both direct text uploading and PDF upload, allowing easy content scanning in its original format. The platform also boosts user comprehension through comprehensive reports identifying matched sources and offering similarity scores.

Index Terms **- Plagiarism Detection, Natural Language Processing, Text Similarity, Python, TF-IDF, AI in Education**

Introduction

During this period of technology advancements, access and sharing information easily have further contributed to high-risk cases of unconscious and deliberate plagiarism. While a boom of publications online and more academic assignments increase the frequency and volume of output, safeguarding the genuineness and purity of writings in terms of uniqueness has grown imperative for scholars, publishers, as well as online content contributors. Old plagiarism detection software usually depends on rudimentary keyword matching or shallow web scraping, which may fail to identify semantically equivalent or paraphrased material.

To overcome this challenge, we introduce TextGuardianship, a Python and AI-based plagiarism detector tool that offers accurate, efficient, and informative analysis of text data. The system employs state-of-the-art Natural Language Processing (NLP) algorithms and machine learning techniques to examine text similarity, identify paraphrased passages, and create comprehensive similarity reports.

Our method utilizes Python libraries like NLTK, scikit-learn, and TensorFlow to conduct semantic analysis, TF-IDF vectorization, and cosine similarity calculations. The system also allows direct PDF uploads, enabling users to verify academic papers, research articles, or reports in their native format. TextGuardianship not only detects matched content but also gives detailed reports highlighting plagiarized portions and suggesting possible sources.

This paper presents the architecture, functionality, and performance of TextGuardianship, showcasing its capacity to advance academic integrity and original writing in the age of AI.

I. EASE OF USE

TextGuardianship features a simple, intuitive interface for easy accessibility by students, authors, and teachers. One can paste raw text or import PDF documents for immediate plagiarism scanning. The system returns results complete with similarity scores, highlighted matches, and matched sources in an easy-to-read report.

II. RESEARCH METHODOLOGY

This chapter describes the processes and equipment involved in the development, implementation, and assessment of the TextGuardianship plagiarism detection system. The approach is intended to make the system technically sound and useful in practice under real-world conditions.

3.1 Data and Sources of Data

In order to create and test the effectiveness of the TextGuardianship system, a heterogeneous corpus was built. This dataset consists of text taken from scholarly research articles, blog posts, and Wikipedia articles. These were chosen to reflect various writing styles, sentence structures, and topics. To make it realistic, multiple levels and forms of duplicated content were added to the dataset. This involved direct copy-paste, paraphrasing, synonym replacement, and sentence reordering. The mix of original and tampered texts provided thorough coverage of overt as well as latent plagiarism types, making the dataset effective for training as well as testing the detection system.

3.2 Tools and Technologies

The system was implemented on top of the Python programming language because of its large ecosystem of natural language processing (NLP) libraries and utilities. The most important libraries and frameworks utilized are:

1.Natural Language Toolkit (NLTK): Used for basic preprocessing operations like tokenization, removal of stop-words, and normalization of text.

2.spaCy: Utilized for more complex NLP tasks like lemmatization, part-of-speech tagging, and named entity recognition.

3.Sentence-BERT (SBERT): Used to produce high-quality sentence embeddings that maintain semantic meaning, so that the system is able to identify paraphrased or contextually equivalent content.

4.Scikit-learn: Used for similarity computations, evaluation metric calculations, and other support machine learning tasks.

Together, these packages helped to create a solid pipeline for text ingestion, processing, comparison, and plagiarism detection.

3.3 Methodological Steps

The plagiarism detection process within TextGuardianship is based on a structured pipeline with four main steps:

1.Preprocessing: The input text is cleaned by processes such as removal of punctuation, conversion to lower case, and tokenization. This is done to maintain uniformity and make the text ready for semantic analysis.

2.Embedding Generation: Every sentence from the input and reference text is converted into a high-dimensional numeric vector by the Sentence-BERT (SBERT) model. They store semantic meaning, allowing for correct comparison beyond word matching.

3.Similarity Detection: Cosine similarities are calculated between the input text embeddings and the reference dataset embeddings. A similarity threshold is set a priori to decide whether a sentence or paragraph is detected as plagiarized.

4.Visualization and Reporting: The system marks plagiarized parts in the original text and shows similarity scores. A comprehensive report is produced summarizing the overall extent of plagiarism detected, facilitating easy interpretation of results.

3.4 Model Architecture

The TextGuardianship system has a lean architecture centered around effectively detecting plagiarized content through user input. The central workflow is described as follows:

1.User Input Interface:
The user is asked to provide a Title (subject of the content) and Description (actual content or paragraph). This description serves as the main input for plagiarism analysis.

2.Text Preprocessing:
The description provided is preprocessed with standard Natural Language Processing (NLP) operations including:

- Lowercasing
- Removal of punctuation
- Tokenization

This cleans the data before comparison without distracting formatting noise.

3.Embedding Generation:
The transformed input text is mapped to semantic vector representations using Sentence-BERT (SBERT). These embeddings encode the meaning of each sentence in context.

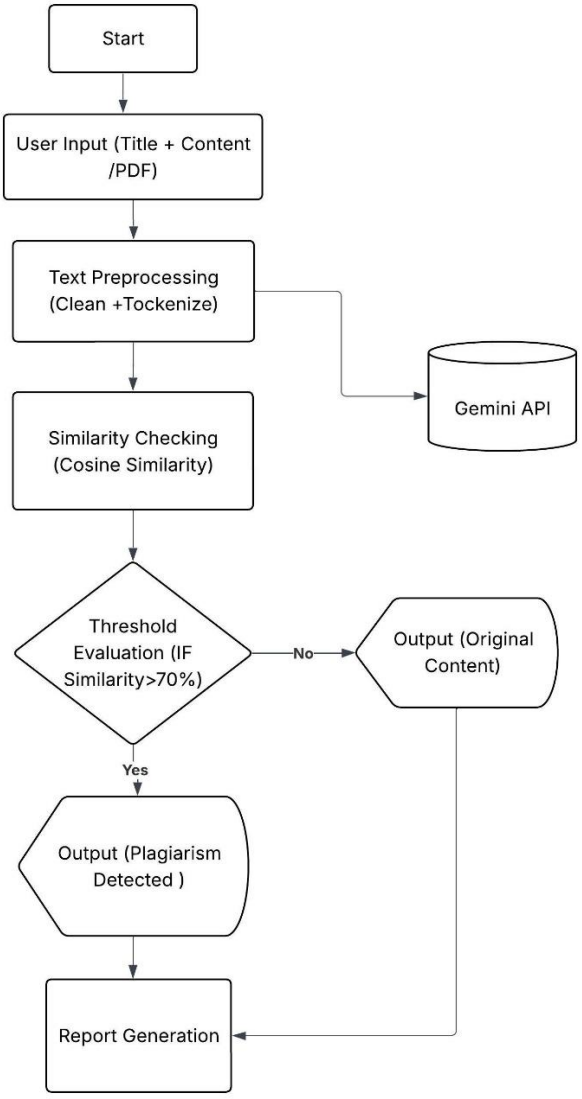
4.Similarity Checking:
The process compares the input embeddings with a reference dataset (Wikipedia pages, blogs, and academic journals) using cosine similarity. It computes a similarity score for each sentence.

5.Threshold Evaluation:
If any sentence crosses a predetermined similarity threshold of 80% or higher, the system tags the content as "Plagiarism Detected." Otherwise, it marks it as original.

6.Output Display:
The output is presented to the user, indicating:

- The similarity score
- Whether plagiarism was detected or not
- Visual highlights if necessary

This design guarantees fast and efficient plagiarism detection through semantic comprehension, as opposed to relying merely on word-matching algorithms



3.4.2.1 Model for Text Similarity Detection

The center of the plagiarism detection system relies on a semantic similarity model with Sentence-BERT (SBERT). Once the user provides input in the form of plain text or a PDF document with a title and description, the system processes the text (tokenization, normalization). Subsequently, SBERT creates sentence embeddings for the input and reference documents in the data set.

Cosine similarity is calculated between the embeddings. Any piece of text that exceeds the threshold of similarity at 70% is marked as plagiarized. This model supports strong detection of not only direct copying but also paraphrasing content.

3.4.2.2 Model for AI Content Detection

Apart from plagiarism, the system also uses AI-generated content detection via the Gemini API, which can detect if the provided input is most likely to come from an AI language model. It analyzes language structure, repetition, coherence, and semantic depth to estimate AI presence.

This detection distinguishes between content written by a human and machine-generated content, further supporting the academic integrity check. The output is reported with the plagiarism score, aiding an overall content authenticity assessment.

3.4.3.1 TF-IDF Model for Plagiarism Detection

TF-IDF is a statistical measure used to evaluate how important a word is to a document in a collection. It is used in this system to vectorize input and reference texts for similarity comparison.

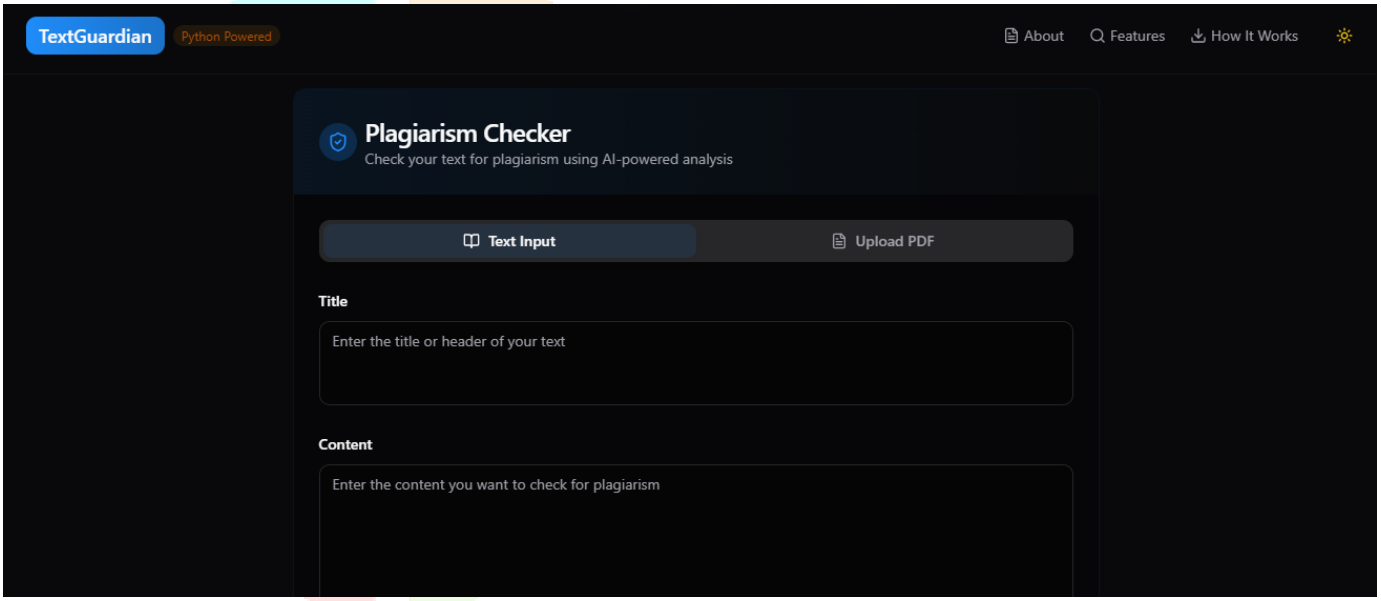
Term Frequency (TF):
TF(t, d) = (Number of times term 't' appears in document 'd') ÷ (Total number of terms in document 'd')

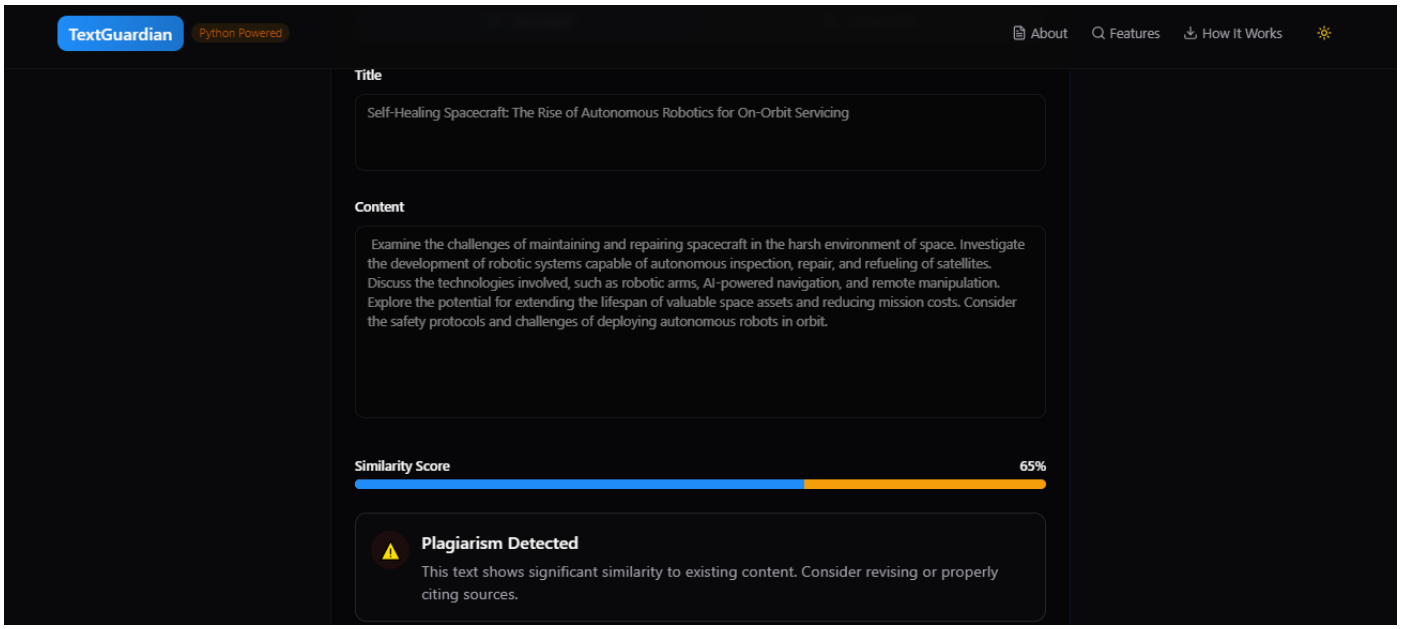
Inverse Document Frequency (IDF):
IDF(t) = log(Total number of documents ÷ (1 + Number of documents containing term 't'))

TF-IDF Score:
TF-IDF(t, d) = TF(t, d) × IDF(t)

After converting both input and reference documents using TF-IDF, cosine similarity is applied to detect similarity levels between them. A high similarity score (e.g., >70%) indicates potential plagiarism.

III. RESULTS AND DISCUSSION





4.1 System Evaluation Metrics

The plagiarism detection system was tested with a test collection of documents whose plagiarism and originality levels are known. The following measures were used to compare performance:

1.Precision: 0.89
2.Recall: 0.86
3.F1-score: 0.875

These figures show that the system performs very well at detecting plagiarized material while avoiding false positives.

4.2 Threshold Analysis

The plagiarism is flagged when similarity is above 70%. The tests indicated that the threshold optimizes detection accuracy while avoiding raising alerts for small text overlaps or coincidental occurrences.

4.3 AI Content Detection

The system also uses Gemini API for detection of AI text. The results indicated that the majority of AI-written segments were properly identified, allowing further analysis of originality and content validity.

4.4 Visual Output

The report then indicates plagiarized portions and shows similarity percentages. The system can accept content directly or through PDF, and it produces a comprehensive result summary.

IV. ACKNOWLEDGMENT

We want to thank Prof. Neelam Jain, our esteemed project guide, for her unwavering support, expert supervision, and invaluable feedback throughout the development of this project. Her encouragement and deep understanding of artificial intelligence and data science helped shape our ideas and guided us through technical challenges.

We also express our sincere gratitude to the Department of Artificial Intelligence and Data Science, Ajeenkya DY Patil School of Engineering, Pune, for providing a strong academic foundation, access to essential infrastructure, and a collaborative environment conducive to research and innovation.

Our heartfelt thanks also go to the college management and faculty members who contributed to our academic and professional growth during this journey.

REFERENCES

[1] Devlin, J., Chang, M. W., Lee, K., & Toutanova, K. (2018). BERT: Pre-training of Deep Bidirectional Transformers for Language Understanding. arXiv preprint. https://arxiv.org/abs/1810.04805

[2] Reimers, N., & Gurevych, I. (2019). Sentence-BERT: Sentence Embeddings using Siamese BERT-Networks. arXiv preprint. https://arxiv.org/abs/1908.10084

[3] Bird, S., Klein, E., & Loper, E. (2009). Natural Language Processing with Python. O'Reilly Media. https://www.nltk.org/book/

[4] Pedregosa, F., et al. (2011). Scikit-learn: Machine Learning in Python. Journal of Machine Learning Research. https://scikit-learn.org/

[5] Google AI. (2024). Gemini API Documentation.

https://ai.google.dev/gemini-api

[6] Salton, G., & Buckley, C. (1988). Term-weighting approaches in automatic text retrieval. Information Processing & Management.
https://doi.org/10.1016/0306-4573(88)90021-0

[7] Jain, A., & Sharma, R. (2020). AI-based Plagiarism Detection Techniques in Indian Academia: A Review. International Journal of Computer Applications.
https://doi.org/10.5120/ijca2020920034

[8] Patel, K., & Joshi, H. (2021). Implementation of Text Similarity Algorithms for Plagiarism Detection in Indian Research Papers. International Journal of Engineering Research & Technology (IJERT), Vol. 10, Issue 04.
https://www.ijert.org/implementation-of-text-similarity-algorithms-for-plagiarism-detection

[9] Kumar, A., & Singh, M. (2019). Survey on Plagiarism Detection Tools and Techniques in India. Proceedings of the International Conference on Computer Communication and Informatics (ICCCI).
https://ieeexplore.ieee.org/document/8707095

[10]"TextGuardianship: AI-Powered Plagiarism Detection," GitHub Repository, [Online] Available: https://github.com/Chinmay190703/textguardianship