**International Journal of Computational Intelligence in Engineering (IJCIE)**
Vol. 01, No. 02, April 2026, pp. 01-16
ISSN: XXXX-XXXX, DOI: 10.5281/zenodo.19554540

# Plagiarism Detection System using Python with Text Similarity Analysis and Result Visualization

**Gangothri C L[1], Chandrappa S[2*]**

[1]Department of CSE (Data Science), Nagarjuna College of Engineering and Technology, Bangalore, Karnataka, India. Email: gangothril65@gmail.com

[2]Department of CSE (Data Science), Nagarjuna College of Engineering and Technology, Bangalore, Karnataka, India. Email: chandrucs21@gmail.com

**Article Info**

***Article history:***

Received Feb 02, 2026
Revised Mar 29, 2026
Accepted Apr 08, 2026

***Keywords:***

Plagiarism detection
Natural language processing
Cosine similarity
Text similarity analysis
Document vectorization

**ABSTRACT**

To provide an effective plagiarism detection mechanism in the ever-growing amount of digital content in academia, research, and business environments, a highly developed need exists for reliable plagiarism detection mechanisms. An integrated plagiarism detection technique combining natural language processing (NLP) methodologies with statistical similarity metrics to detect similar content within many different documents utilizing a python based platform is presented herein. All input documents are first pre-processed by means of tokenizing the input texts and removing stop-words to obtain a reduced set of only meaningful tokens. Each document is then transformed into a numeric vector using the TF-IDF methodology which emphasizes uniquely occurring terms within each document and diminishes commonality among words in the other documents. Pairwise cosine similarity is then computed for all combinations of documents resulting in a similarity matrix where each entry represents the degree of similarity between the corresponding pair of documents. Five documents representing three disciplines were utilized to test this technique. High similarity scores were reported (0.7) between documents one and two which represent the same content regarding machine learning, indicating that they represented plagiarism cases. In contrast low-similarity scores were reported for unrelated document pairs, greatly reducing false positives. Moderate similarities (0.55) were also reported for two related climate science documents; these values indicate some degree of overlap but no direct plagiarism.



***Corresponding Author:***

Chandrappa S
Department of CSE (Data Science), Nagarjuna College of Engineering and Technology, Bangalore, Karnataka, India.
Email: chandrucs21@gmail.com

## 1. INTRODUCTION

The Plagiarism has become one of the biggest challenges of our times due to rapid advancement of digital contents in all types of media; academic, professional, and online. There are many publicly accessible tools today using Large Language Models (LLMs) that generate content that make plagiarism very common among students today because of the easy access to vast amounts of electronically generated information on the Internet [1]. Not only does it affect the validity of academic work but it also affects both ethics and legality for individuals and institutions. According to studies, as of 2023, up to 58% of University students reported participating in some type of plagiarism throughout their academic careers. Also, an estimated 1.5% of all published papers contain duplicated content or reused unethical material [2]. Therefore, there has been much interest in developing plagiarism detection systems with automatic, accurate, and quick efficiency over the last few years [3].

Historical plagiarism detection processes have mainly consisted of manual inspections which are both time consuming and subject to human error. Early plagiarism detection methods using string matching algorithms were successful at identifying plagiarism that was direct copies of text. Tools such as Turnitin and CopyCatch utilize Rabin-Karp and Knuth-Morris-Pratt algorithms to find overlapping sections of text when comparing different sections of text [4]. Over the past decade however, plagiarism has progressed far beyond simply copying a section of text and includes paraphrasing, translating ideas into other languages, paraphrasing, and even generating content using Artificial Intelligence (AI). These newer forms of plagiarism have proven to be difficult for traditional string matching algorithms to catch [5]. To address this issue, improvements in both natural language processing (NLP) and machine learning have greatly improved plagiarism detection capabilities by introducing semantic similarity models, deep neural network architectures, and statistical text representations [6].

TF-IDF (term frequency-inverse document frequency) vectorization along with cosine similarity have emerged as the top two methods for determining how similar two texts are. After converting the input text into lowercase tokens, the TF-IDF value represents the relative importance of each token within the input document and is calculated by multiplying the term frequency by the inverse document frequency [7]. Cosine similarity then calculates the cosine of the angle between two high dimensional document vectors, resulting in a scaled similarity measure between 0 and 1 that indicates how close two texts are in relation to one another [8]. Both TF-IDF and cosine similarity have been demonstrated in numerous studies to be viable for plagiarism detection purposes [9]. Similarity-based detection requires proper pre-processing before the similarity detection can take place. A typical NLP pipeline consists of pre-processing, tokenizing, and additional analytical steps. Pre-processing is comprised of tasks that include removing punctuation and special characters, converting text to lowercase, lemmatizing words to base forms, and removing stopwords – semantically irrelevant words such as “a”, “an”, “the”, etc., which add noise to unstructured text. Removing stopwords has been demonstrated to produce a higher signal-to-noise ratio in unstructured text and significantly increases the relevance of terms being compared in similarity detection [11]. Tokenizing converts raw unstructured text into structured entities that can be extracted into downstream feature extractions and vector representations [12] – collectively providing the foundational building blocks for robust similarity-detection pipelines [13].

Although numerous commercially available plagiarism detection tools exist today, most current plagiarism detection tools function as “black boxes”, providing minimal insight into how they compute similarities between documents [14]. Additionally, most commercial plagiarism detection tools require substantial financial investments and do not provide visualizations of similarities across multiple documents [15]. Recently, numerous studies have emphasized the need for transparently understandable, open-source plagiarism detection toolsets that researchers and educators can easily interpret and use [16]. An extensive comparative study of embedding approaches confirmed that the classical TF-IDF method outperformed several modern embedding methods in detecting similarities at the document level confirming its continued utility in plagiarism-detection workflows [17]. Researchers recently explored an array of alternative plagiarism-detection methods outside of TF-IDF; including Deep Learning-based models such as BERT (Bidirectional Encoder Representations from Transformers), Long Short-Term Memory (LSTM) Networks, Transformer Architectures; all of which perform exceptionally well at detecting paraphrase-style plagiarism [18][19]. Contextualized word embeddings (e.g., Word2Vec, GloVe); and Transformers utilizing Generative Pre-training Transformer (GPT)-based architectures enable nuanced detections by representing contextualized word meanings. Similarity is often computed via cosine similarity within embedding spaces as well [20]. Numerous Support Vector Machine (SVMs) and Random Forest Classifier implementations trained on Lexical-Syntactic-Semantic attributes have achieved remarkable accuracy levels (up to 92%) in evaluating plagiarism [21]. Semantic Role Labeling (SRL), Named Entity Recognition (NER), and other concept-based plagiarism-detection methods have expanded detection capabilities to detect plagiarism beyond superficial textual overlap; i.e., at the idea level [22].

Visualizing similarity results is another aspect currently lacking in many plagiarism detection systems. Heatmaps created from similarity matrices offer a highly intuitive way for users to visualize inter-document relationships and rapidly identify suspicious document pairs without manually inspecting each pair's similarity scores [23]. Automatically flagging suspect document pairs based on a defined similarity threshold reduces the amount of time required by end-users to evaluate detected instances of plagiarism [24]. The integration of Statistical-NLP Modeling with Visual Analytics offers an efficient, interpretable, and scalable framework for Educational Institutions, Content Platforms and Research Organizations looking to maintain content integrity [25]. In addressing the aforementioned shortcomings this work introduces a novel Python-based Plagiarism

Detection Framework that utilizes TF-IDF vectorization, cosine-similarity computations and graphical visualization through heatmaps. The proposed framework accepts input from multiple documents, preprocesses them via tokenization & Stopword removal; produces a similarity matrix between all possible combinations of documents; and automatically identifies and reports all document pairs exceeding a user-defined similarity threshold. Results are displayed graphically as a color-coded heatmap, allowing users to instantly identify suspicious document pairs. The remainder of this paper is organized as follows. Section 2 reviews relevant literature associated with plagiarism-detection and text-similarity evaluation. Section 3 describes the Architecture and Methodology utilized in the Proposed System. Section 4 details Experimental results and Analysis. Section 5 summarizes the findings of this study and provides suggestions for Future Study.

## 2. LITERATURE REVIEW

This chapter reviews the body of existing research on plagiarism detection, text similarity analysis and other related Natural Language Processing (NLP) techniques. The systematic literature surveys of [1][2], focused on the details of feature set engineering and similarity computation techniques employed for plagiarism detection classification, are well-documented. Early plagiarism detection systems mainly employed exact string matching and rule based algorithms. In particular, Parwita et al. (2019)[4]proposed a plagiarism detection system for documents written in Bahasa indonesia employing the Rabin-Karp rolling hash algorithm to detect verbatim overlap between documents. Although the use of exact string matching proved successful for direct copy/paste plagiarism, this approach was also shown to be less effective when dealing with paraphrased and/or restructured versions of the original content. In addition, Foltýnek et al. (2019)[15]conducted a comprehensive systematic literature review of academic plagiarism detection spanning several decades; they concluded that, although computationally inexpensive, string matching methods are incapable of detecting semantic level plagiarism, where the surface form of the text has been modified to preserve the meaning. Thus, there is a need for plagiarism detection methods that employ more linguistically informed methods. Chaubey and Chaubey (2022)[5]conducted a comparative study of automatic plagiarism detection methods on multilingual corpora. They reported that hybrid feature extraction methods that combine both surface level and semantic features achieved higher levels of accuracy than methods employing single feature string matching alone. TF-idf along with cosine similarity represent one of the most popular statistical approaches in the area of text-based plagiarism detection. Halim and Lasut (2024)[7]developed a web-based plagiarism detection tool employing TF-idf and cosine similarity, whereby each document was pre-processed by means of tokenization and stop-word elimination prior to the conversion to weighted term vectors. Experimental results indicated that the tool effectively detected similarities between English-language academic documents. Riyani et al. (2019)[17]further supported this approach by applying TF-idf weighting and cosine similarity to Indonesian documents, indicating that cosine similarity provided a reliable and language independent measure of document overlap. Husain et al. (2024)[9]extended this work by developing a plagiarism detection tool employing cosine similarity to determine whether pairs of electronic documents exhibited similar content. Benchmarking experiments were carried out using manually annotated corpora and reasonable accuracy levels were obtained across different document formats. Sari (2023)[8]applied TF-idf and cosine similarity to student thesis documents and demonstrated that a similarity threshold value greater than 80% could be used to reject suspected plagiarized submissions to an academic repository. Tokenization and stop-word removal in the preprocessing phase were demonstrated to improve input data quality resulting in more accurate TF-idf calculations. A comparative evaluation conducted by Husain and Suryani (2022)[21]examined svm and rf classifiers trained on TF-idf derived feature vectors for text-based plagiarism detection and reported accuracy levels of up to 92%; these results support the widespread adoption of TF-idf as a source of feature-extraction data. Several studies have recognized that text preprocessing is crucial in determining the quality and accuracy of similarity-based plagiarism detection methods. Pichiyan et al. (2023)[10]studied the combination of web scraping and NLP methods for unstructured text analysis and showed that tokenization, stemming, and stop word removal are important preprocessing steps that greatly affect the efficacy of subsequent similarity computations. Sarica and Luo (2021)[11]carried out an empirical investigation of the importance of stop-word removal in technical language processing and showed that removing high-frequency words that carry little semantic information increases the signal-to-noise ratio in text collections and enhances the accuracy of NLP-based analytical tasks. The authors created a domain-specific stopwords list based on term frequency, inverse document frequency, and entropy metrics. Kaur and Sohal (2024)[12]also examined noise estimation and reduction in NLP pipelines and confirmed that structured preprocessing pipelines consisting of tokenization, lowercase transformation, punctuation removal, and stop-

word removal are necessary components of reliable text similarity systems. Together, these preprocessing steps form the foundation upon which all robust text similarity pipelines are built.[13]

A number of researchers have studied the application of supervised machine learning classifiers to plagiarism detection. For example, El-Rashidy et al. (2024)[16]developed a three stage plagiarism detection pipeline comprising part-of-speech tagging, lemmatization, and stop-word removal as preprocessing steps. Then they extracted lexical, syntactic, and semantic features from the processed tokens prior to training a classifier using an svm algorithm. Their experimental results show that their approach achieves better performance than conventional string matching baselines. In a separate study Foltýnek et al. (2024)[1]carried out a systematic review of 189 research articles published between 2019 and 2024 assessing various plagiarism detection techniques; they reported that machine-learning-based methods out-performed rule-based methods in terms of detecting paraphrased or semantically transformed content. However, the review highlighted that feature-selection is key to ensuring effective classification performance. In another study Gandhi et al. (2024)[3]reviewed AI-based plagiarism detection tools; they assessed their ability to identify duplicate text, recognize writing-style similarities, and detect cases of self-plagiarism across different types of content. Based on their assessment, they concluded that collective employment of keyword-analysis, text-pattern matching, and semantic analysis using NLP techniques will lead to increased accuracy in detecting plagiarized content.

More recently, the focus has shifted towards deep learning architectures capable of capturing contextual semantic relationships between texts. Bohra and Barwar (2022)[18]proposed a plagiarism detection algorithm using bert (bidirectional encoder representations from transformers); the proposed method generated contextualized word embeddings from a pre-trained bert model and computed a sentence similarity metric for comparing suspicious content against a collection of references; they claimed improved efficiency relative to traditional plagiarism detection methods. Rosu et al. (2021)[6]presented an NLP-based deep-learning approach for plagiarism detection utilizing roberta and sentence-transformer-based networks; they validated their approach on a dataset containing eleven medium-length documents and demonstrated good discrimination ability between sentences describing content with varying topical closeness.

Moravvej et al. (2022)[19]proposed an optimized de (Differential Evolution) algorithm for enhancing the training speed of bert-based plagiarism detection models; their approach leveraged evolutionary computing techniques to achieve faster optimization times. Hayawi et al. (2023)[25]employed lstm-based deep learning models for paraphrasing-level plagiarism detection; their experiments demonstrated that sequential neural network architectures are particularly adept at identifying subtle textual transformations that evade traditional similarity measures. An examination of different embedding-methods revealed by Riyani et al. (2019)[17]and extended by Husain et al. (2024)[9]found that traditional TF-idf method surpassed many modern embedding-methods in detecting document-level similarity; hence, this supports its ongoing viability in conjunction with transformer-based approaches in complete plagiarism detection workflows. Word embedding models have proven themselves useful for identifying semantic relationships between words and documents exceeding surface-level term matching. Mehak et al. (2023)[20]carried out a contextual analysis examining bert/gpt-based transformer embeddings for plagiarism detection; they discovered that contextual embeddings far surpass static word-vectors like word2vec/GloVe in detecting paraphrased content.

Mokoatle et al. (2023)[27]evaluated SBERT (Sentence-BERT)/simcse embedding-models for semantic-similarity-detection; they validated that sentence-level embeddings produce richer representations for document-comparison than word-level embeddings. Latina et al. (2024)[23]utilized word2vec/bert in an NLP-based plagiarism detection system; they applied semantic-analysis to identify similarities missed by traditional keyword-matching approaches. Beyond lexical/semantic-similarity-detection methods concept-based detection methods have emerged to find idea-level plagiarisms where content is substantially restructured. Taufiq et al.,(2023)[22]proposed a concept-based plagiarism detection approach using semantic role labeling(named entity recognition), combining syntactic & semantic measures through Wu-Palmer similarity to detect deeper textual relationships missed by word-level analysis. Meuschke et al., (2018)[30]propose an adaptive image-based plagiarism detection method extending beyond text-based detection to structural/visual elements of academic documents. Foltýnˇek et al.,(2019)[15]further categorized advanced approaches within a broader taxonomy of plagiarism types including verbatim plagiarism, paraphrasing, translation and idea-based plagiarism as successively more challenging detection targets.

The emergence of large language models such as ChatGPT & Gemini has introduced new challenges for plagiarism detection systems. Guo et al.(2023)[26]were among first to compare systematically human-written and ChatGPT-generated texts; they showed that existing detectors trained on datasets written by humans struggle to reliably identify AI-generated content from original writing. Foltýnˇek et al.,(2024)[1]confirmed growing concern regarding AI-generated content in comparative study review published between 2019 and 2024; survey conducted in same study reported up to 58% university students admitted engaging some form of plagiarism during their academic careers; hence, there is need for reliable automated detection tools. Often over-looked aspect of research in area of plagiarism detection is presentation and interpretation of results. Latinas et al.,(2024)[23]showed that scores and visualizations are important outputs for presenting results clearly to end users especially in educational setting. Al-Qura'an et al.,(2025)[13]proposed threshold-based flagging mechanism which provides proportionate scaling of similarity values to percentage range of 0% to 100% for intuitive interpretation; similarly proposed web-based tool employing tfidf/cosine-similarity utilizing online resources via web scraping; also emphasized importance of clear visualization and numerical output representation requirements for practical deployment of plagiarism detection tools in academia; presented heatmap-based visualization alongside numerical similarity score representation as part of proposed tool.

Table 1. Comparative analysis of recent works in the research field

| Ref | Year | Method | Dataset | Key Contribution | Limitation |
|---|---|---|---|---|---|
| **[8]** | 2023 | TF-IDE | Student thesis repository | 80% similarity threshold shown effective for rejecting plagiarized academic submissions | Threshold may vary across domains; lacks deep semantic analysis |
| **[10]** | 2023 | NLP + Web Scraping (Tokenization, Stemming) | Unstructured web text | Tokenization, stemming, and stop word removal improve downstream similarity computation | Focused on extraction only; no end-to-end plagiarism detection pipeline |
| **[20]** | 2023 | BERT + GPT Transformer Embeddings | Paraphrase datasets | Contextual embeddings substantially outperform Word2Vec/GloVe for paraphrase detection | High resource requirements; limited interpretability for end users |
| **[22]** | 2023 | Semantic Role Labeling (SRL) + NER + Wu-Palmer | Structured academic text | Concept-based detection identifies idea-level plagiarism beyond word overlap | Complex pipeline; high computational cost; depends on NLP parsers |
| **[25]** | 2023 | LSTM Deep Learning Models | Paraphrase-level corpus | Sequential neural models capture subtle textual transformations that evade traditional similarity measures | Weak generalization to concept-level or translation plagiarism |
| **[26]** | 2023 | Statistical + ML Detection Classifiers | ChatGPT vs. human-written texts | Demonstrated existing detectors struggle to distinguish AI-generated from human-authored content | Rapidly evolving LLMs reduce detector reliability over time |
| **[27]** | 2023 | SBERT + SimCSE Sentence Embeddings | Semantic similarity benchmarks | Sentence-level embeddings provide richer document representations than word-level models | Performance degrades on highly domain-specific or technical corpora |
| **[28]** | 2023 | Decision Tree Classifiers | Text classification datasets | Decision trees offer interpretable classification for text similarity tasks | Lower accuracy compared to deep learning methods on complex paraphrases |
| **[29]** | 2023 | All-MiniLM Embeddings + Wikipedia Data | Wikipedia readability corpus | Lightweight sentence embeddings achieve competitive classification with reduced compute | Limited to readability tasks; not directly validated on plagiarism datasets |
| **[1]** | 2024 | Comparative Review (189 papers, 2019–2024) | Multi-paper corpus | ML-based methods outperform rule-based; AI-generated content is the most critical emerging challenge | Review only; no new detection system proposed |
| **[3]** | 2024 | AI-Powered Detection Tools Review | Multiple content types | NLP techniques combining keyword analysis, pattern matching, and semantic analysis improve | Evaluation of commercial tools; limited reproducibility of findings |

| | | | | | |
|---|---|---|---|---|---|
| | | | | accuracy | |
| **[7]** | 2024 | TF-IDF + Cosine Similarity (Web-based app) | English academic texts | Web-based system with tokenization and stop word removal; effective for document-level similarity | No sentence-level granularity; limited to English-language documents |
| **[9]** | 2024 | Cosine Similarity (Electronic Documents) | Ground-truth annotated corpus | Benchmarked cosine similarity across multiple document types with reliable detection performance | Fails on heavily paraphrased content; no semantic understanding |
| **[14]** | 2024 | Deep Learning + NLP (Text + Image) | Multi-modal academic documents | Integrated text and image plagiarism detection with heatmap-based visualization output | Requires large annotated datasets; computationally expensive |
| **[16]** | 2024 | SVM + Lexical/Syntactic/Semantic Features | Academic text corpus | Three-stage pipeline (POS tagging, lemmatization, feature fusion) outperforms string-matching baselines | High preprocessing overhead; requires labelled training data |
| **[23]** | 2024 | Word2Vec + BERT (Semantic Analysis) | NLP benchmark datasets | Semantic analysis detects similarities that conventional keyword matching fails to identify | Struggles with heavily restructured or translated content |
| **[24]** | 2024 | Deep Learning Systematic Review | Multi-paper review corpus | Systematic survey of deep learning approaches; identifies gaps in paraphrase-level detection | Review only; no unified benchmark for cross-method comparison |
| **[2]** | 2025 | Systematic Survey (Plagiarism Types + Algorithms) | Multi-paper survey | Comprehensive survey of plagiarism types and detection algorithms covering text analysis methods | Survey scope; implementation and real-world validation not covered |
| **[12]** | 2024 | NLP Preprocessing Pipeline (Noise Removal) | NLP pipeline benchmarks | Confirms tokenization, lowercasing, punctuation removal, and stop word filtering as foundational steps | Focused on preprocessing only; no standalone plagiarism detection evaluation |
| **[13]** | 2025 | TF-IDF + Cosine Similarity + Web Scraping | Online academic submissions | Threshold-based flagging with 0–100% scaled similarity scores for intuitive interpretation | Web scraping dependency may limit real-time deployment performance |

## 3. METHODOLOGY

Beginning with the dataset description, this segment provides information about the dataset utilized by the system, its preprocessing pipeline, system architecture, methodology proposed, algorithms for calculating similarities and the visualizations employed in the Python based Plagiarism Detection System. All parts of the system are written exclusively in Python and do not rely upon large annotated training datasets nor specialized hardware.

### 3.1 Dataset

The system can accept plain text files as input and relies upon no specific external dataset. To allow for both experimental evaluation/ validation of the system, a collection of five text documents were compiled from three diverse subject areas:

• Artificial Intelligence/Machine Learning

• Tropical Ecology (Amazon Rain Forest)

• Climate Science

The multi-domain set of documents were designed to test the ability of the system to identify plagiarized documents that have similar content to each other within their respective domains (while minimizing false positive results among documents that cover semantically unrelated subjects). Documents 1 & 2 include

substantially similar content regarding machine learning topics; Documents 4 & 5 present similar climate-related issues; and Document 3 discusses a completely different ecological area of study.

### 3.2 Data Pre-Processing

Noise in the raw text data is a major challenge to determining text-based similarity. Therefore pre-processing occurs on all input documents before any similarity calculations occur. The data-preprocessing pipeline includes the following sequential steps:

Lowercase conversions are made to all input text so that if the same word appears in different case types (i.e., "Machine" and "machine"), it will be treated as a single token. Lowercasing standardizes the vocabulary and helps reduce artificial dimensionality in the term vector space.

#### 3.2.2 Punctuation/Special Characters Removal

Punctuation marks/symbols, special characters and non-alphanumeric symbols are removed from all text. As an example, having two separate tokens "learning." and "learning" creates unnecessary noise in the TF-IDF vector representations.

#### 3.2.3 Tokenization

Tokenization involves breaking down clean text into individual words/tokens. Each token is treated as a unique term during the feature-extraction phase. Whitespace separation is employed to tokenize all text, subsequent to applying lowercase conversion and removing punctuation [10].

#### 3.2.4 Stop Word Removal

High frequency but semantically uninformative terms including "the," "is," "a," "and," "of," "in" etc. appear in nearly every document and contribute very little when attempting to distinguish between documents. Thus, removal of stop words greatly minimizes noise in addition to enhancing the discriminative characteristics of the TF-IDF vectors [11]. Standard English stop word lists are employed to remove all stop words from each tokenized document.

### 3.3 Proposed Methodology

Plagiarism detection employs a modular-pipeline based methodology where the entire workflow is defined by Algorithm 1 below. It illustrates the sequence of operations from receiving raw document inputs through to finally producing plagiarism detection decisions based on computed similarities.

**Algorithm 1: Communication-Efficient Federated Learning for CT Image Classification**

| |
|---|
| **Input:**<br>• Text document corpus $D = \{d_1, d_2, \dots, d_n\}$<br>• Similarity threshold $\tau \in [0, 1]$<br>• Stop word list $SW$ |
| **Output:**<br>• $N \times N$ pairwise cosine similarity matrix $S$<br>• Set of flagged document pairs $F = \{(i, j) \mid S[i][j] \geq \tau, i \neq j\}$<br>• Heatmap and result visualizations |
| **Step 1:** Preprocess all input documents<br>1. Convert all text to lowercase<br>2. Remove punctuation and special characters<br>3. Tokenize each document $d_i$ into token set $T_i$ |

4. Remove stop words: $T_i \leftarrow T_i - SW$

**Step 2:** Apply TF-IDF vectorization

5. Construct shared vocabulary $V$ from all preprocessed documents
6. For each term t and document $d_i$, compute:

$$TF(t, d_i) = freq(t, d_i) / |T_i|$$

$$IDF(t) = log( (N + 1) / (df(t) + 1) ) + 1$$

$$W(t, d_i) = TF(t, d_i) \times IDF(t)$$

7. Represent each document as TF-IDF vector $v_i$ over $V$

**Step 3:** Compute pairwise cosine similarity

8. For each document pair $(i, j)$, compute:

$$CosineSimilarity(A, B) = (A \cdot B) / (||A|| \times ||B||)$$

9. Construct symmetric $N \times N$ similarity matrix $S$ where $S[i][j]$ = cosine similarity of $d_i$ and $d_j$

**Step 4:** Apply threshold-based plagiarism flagging

10. For each pair $(i, j)$ where $i < j$:
11. If $S[i][j] \geq \tau$, add $(i, j)$ to flagged set $F$

**Step 5:** Repeat until all document pairs are evaluated

**Return:** Final global similarity matrix S and flagged set $F$

### 3.4 Feature Extraction - TF-IDF Vectorization

Term Frequency–Inverse Document Frequency (TF-IDF) is a statistical weighting technique used to transform raw-text documents into high-dimensional numeric vectors. TF-IDF computes a weighted value based upon the relevance of each term within a single document, compared to its global frequency across an overall set of documents [7][17].

The Term Frequency (TF) for a given term t within a document d is defined as:

$$TF(t, d) = \frac{Frequency\ of\ term\ t\ within\ document\ d}{(Total\ number\ of\ terms\ in\ document\ d)}$$

The Inverse Document Frequency (IDF) across a total corpus of $N$ documents is:

$$IDF(t) = Log(N + 1)/(df(t) + 1)) + 1$$

Where df(t), is the number of documents in which the term t occurs. A smoothing factor of 1 is included in the addition to both the numerator and denominator, so as to prevent division by zero. Finally, the weighted TF-IDF is:

$$W(t, d) = TF(t, d) * IDF(t)$$

First, a shared vocabulary is created from all unique terms found in all documents after they have been pre-processed. Next, each document will be represented as a vector of TF-IDF weights over this vocabulary. Words that occur uniquely in one document will have high TF-IDF weights; conversely, words that occur in many documents will have lower TF-IDF weights, allowing for differentiation among documents [9].

### 3.5 Similarity Computation - Cosine Similarity

Once all documents are converted to TF-IDF vectors, the cosine similarity between each pair of documents is calculated to create the $N\ x\ N$ similarity matrix. Cosine similarity is used to compute the cosine of the angle between two high-dimensional vectors and is described mathematically as:

$$Cosine\ Similarity(A, B) = (A \bullet B) / (||A|| * ||B||)$$

Where A•B represents the dot-product of vectors A and B and ||A|| and ||B|| represent the Euclidean norm of vectors A and B respectively. Cosine similarity has a range from 0 to 1, where 1 means that the two documents contain identical information and 0 means that there is absolutely no relationship between the two documents. Because cosine similarity does not depend on the size of a document, and thus is able to compare documents regardless of size [8], it is generally considered superior to Euclidean distance for comparing documents.

### 3.6 Threshold Based Plagiarism Flagging

After calculating the similarity matrix, a user specified similarity threshold τ is applied to automatically detect and alert users to potentially plagiarized document pairs. If any pair of documents (i,j) exists such that $i \neq j$ and $S[i][j] \geq \tau$ is detected, the pair is identified as a possible case of plagiarism. For the purpose of experimentation, the authors used a threshold of $\tau=0.50$. This mechanism also allows users to dynamically change the threshold τ in order to increase or decrease the sensitivity of the plagiarism detection system.

### 3.7 Results Visualization

The system includes seven different types of results visualization to provide users with tools to facilitate understanding of results and make informed decisions. Specifically these outputs are:

- Table Displaying Similarity Matrix - the complete N x N pairwise cosine similarity matrix displayed as a tabular format for viewing.
- Colored Annotated Heatmap - colored heat map of actual similarity values placed at each individual cell for simultaneous visual and numerical examination.
- Average Similarity Bar Chart - bar chart displaying the average similarity per document when compared to the entire body of work.
- Document Similarity Network Graph - graph where each node represents a document and edge connections exist if a pair exceeds the previously established threshold.
- Similarity Trend Line Plot - line plot representing sequential average similarity trends from Doc1 through DocN.
- Content Word Frequency Distribution - bar chart representing top content words (after stopword exclusion) influencing similarity scores.
- Pie Chart Representing Plagiarism Distribution - high level representation indicating plagiarism distribution percentages between plagiarized/non-plagiarized document pairs.

### 3.8 System Implementation

The system is written completely in Python version 3 utilizing the following third-party libraries/tools:

1. scikit-learn - TF-IDF vectorization (TfidfVectorizer) and cosine similarity calculation (cosine similarity).
2. pandas - creation/display of similarity matrix as structured pandas DataFrame.
3. matplotlib & seaborn - creation/visualization of heatmaps along with improved graphics using color annotations.
4. networkx - construction of document similarity network graphs.
5. numpy - underlying numerical array-based operation for vector calculations.
6. Google Colab - development/environment execution platform (no need for local installation).

The system is designed to be modular, allowing users to easily expand functionality. Users can add multiple documents, modify thresholds, enable/disable stopword filtering, and switch between TF-IDF/Cosine Similarity/Jaccard Similarity as the primary analytical technique.

## 4. RESULTS AND DISCUSSION

This The experimental results for the Plagiarism Detection System (implemented as an application in Python)

were tested using a set of five documents in total covering three different fields. The test results are described within six separate sections of this chapter; the similarity matrix that the system outputs, heat map visualizations, average document similarity, the document similarity network, the similarity trend analysis, word frequency distributions and plagiarism distributions.

### 4.1. Similarity Matrix Output

Similar to many other systems based on natural language processing that evaluate documents against one another through some form of comparison, the primary output of our system is a 5 × 5 matrix of pairwise cosine similarities. These similarities were calculated using TF-IDF vectorization of the text from each of the five input documents. All entries in this matrix are symmetric, since every document has been compared with every other document. The diagonal entries have values of 1.000000; these represent perfect similarity between each document and its self. A full copy of the similarity matrix generated by our system is provided in Table 4.1.

Table 4.1. Cosine similarity matrix of the five experimental documents.

| | Doc1 | Doc2 | Doc3 | Doc4 | Doc5 |
|---|---|---|---|---|---|
| **Doc1** | 1.000000 | 0.703692 | 0.246636 | 0.261845 | 0.117187 |
| **Doc2** | 0.703692 | 1.000000 | 0.211550 | 0.224596 | 0.100517 |
| **Doc3** | 0.246636 | 0.211550 | 1.000000 | 0.000000 | 0.000000 |
| **Doc4** | 0.261845 | 0.224596 | 0.000000 | 1.000000 | 0.548530 |
| **Doc5** | 0.117187 | 0.100517 | 0.000000 | 0.548530 | 1.000000 |

From As can be seen from the table 4.1, the highest off-diagonal similarity value of 0.703692 exists between Doc1 and Doc2; these two have overlapping information about machine learning and neural networks. In addition, a moderate similarity of 0.548530 was determined for Doc4 and Doc5; these two cover climate-related topics. It is also confirmed that there is no similarity (0.000000) between Doc3 (Amazon Rainforest Ecology), and Doc4/Doc5 because it has been demonstrated that the system could find non-relevant documents accurately without generating false positive results.

### 4.2. Visualizing Similarity Heat Map

The heat map visualization of the similarity matrix is created based on color scale using the Matplotlib and Seaborn libraries. This allows for easier to understand visual comparison of all possible pair-wise similarities among the documents; warm/brighter colors represent high similarity values whereas dark cells represent low similarity. The heat map produced by the system for the experimentally gathered data is shown in Fig. 4.1.

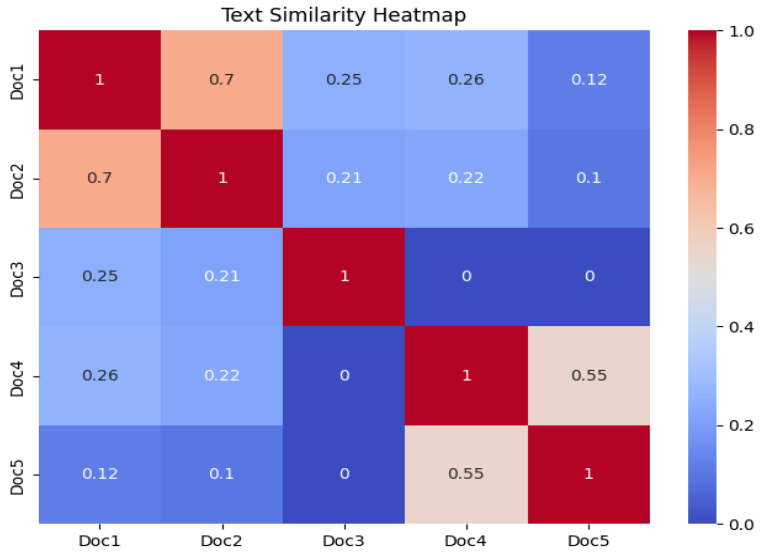


Fig 4.1. Heatmap visualization of the pairwise cosine similarity matrix.

The most brilliant off-diagonal cell that can be seen in figure 4.1 is for the Doc1- Doc2 pair which has an apparent similarity of .70; it will be noticeable from every other pair. A somewhat brighter but less intense cell is also present for the Doc4 – Doc5 pair at .55. Off-diagonal pairs that do not include either of these two have very dark values, indicative of little or no similarity.

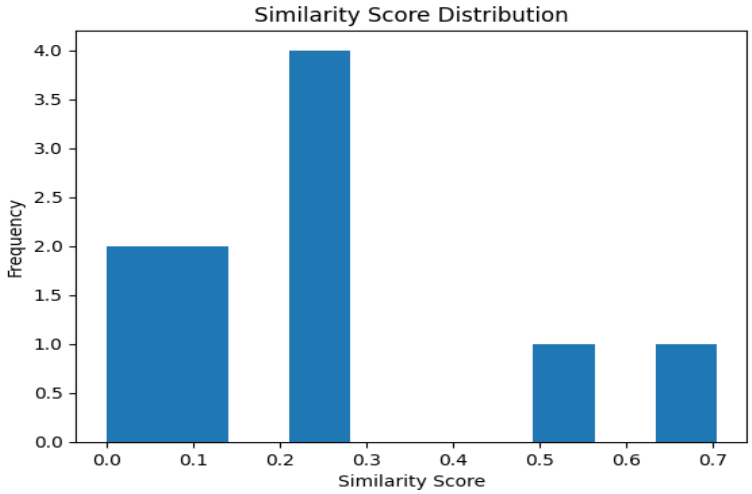


Fig 4.2. Annotated similarity heatmap showing exact score values in each cell.

Figure 4.2 displays an annotated version of the previous heatmap where the actual numerical similarity scores are superimposed on each cell. It enables a detailed examination of borderline similar scores that may be difficult to see using just colors within the heatmap. Thus, it will provide both a visual and numeric representation for further analysis.

### 4.3. Average Similarity per Document

The average cosine similarity score has been calculated for each document relative to the whole corpus. For this purpose, the scores have been averaged for every document with respect to all other five documents (also including the case when two different versions of a document compare themselves). These results are presented as a bar chart in Figure 4.3.

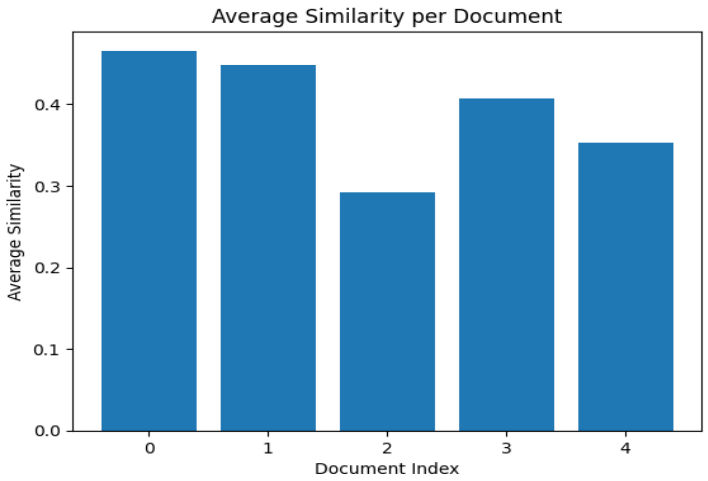


Fig 4.3. Bar chart showing average cosine similarity score per document.

As shown in FIGURE 4.3, Doc1 and Doc2 show the largest average similarities because they are highly similar in both content and the amount of shared information with Doc4. Doc3 has the smallest average similarity value which signifies it is topically unique (Rainforest at Amazon) compared to all other documents. This evaluation also demonstrates that the TF-IDF and cosine similarity based pipeline correctly identifies each document's level of closeness to all other documents within the total collection.

### 4.4. Document Similarity Network Graph.

The NetworkX library was used to generate a document similarity network graph to display the relationships between documents as a graph structure. Each document was depicted as a single node on the graph, but an edge existed between two nodes when the cosine similarity measure between those two nodes exceeded the established similarity threshold of 0.7. The network graph is displayed below in FIGURE 4.4

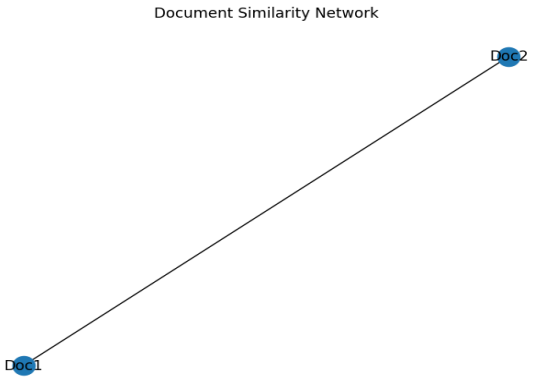


Fig 4.4. Network graph of document similarity connections at threshold 0.70.

As depicted in figure 4.4, there are only two entities connected within the graph; those being Doc1 and doc2 as they are the only documents whose similarities exceed 0.7 or better than the threshold value set within the experimental data. All other documents, i.e., doc3, Doc4 and doc5 are represented solely as single nodes with no connections to either an individual document or another node. Therefore, none of these documents were flagged as similar enough to be included in the plagiarism detection output. The visual presentation of the network graph clearly illustrates the result obtained from the plagiarism detection method.

### 4.5. Plots of document similarities over time.

A line graph showing the average similarity value for each document will provide insight into any trends regarding the way that document similarities may vary over time. These types of plots are useful for illustrating whether or not the degree of similarity amongst the various documents changes over sequential intervals. A sample trend plot is presented in figure 4.5.

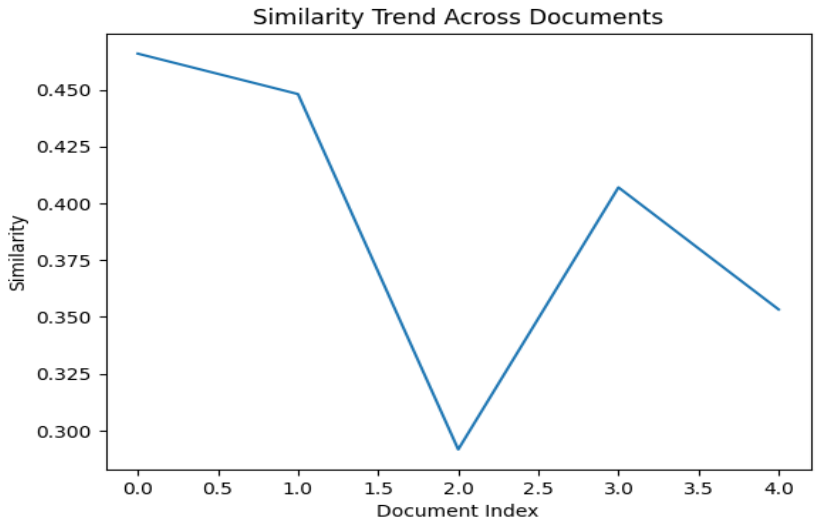


Fig 4.5. Line plot showing similarity trend across documents (Doc1 to Doc5).

Figure 4.5 illustrates that Doc1 and Doc2 show the greatest overall average similarity value levels; they represent an obvious peak in the early part of the sequence. The similarity decreases after Doc3, increases moderately for Doc4 and Doc5 by reason of both having a climate theme, and remain at a significantly lower similarity value than the peak shown by Doc1 and Doc2. This trend analysis offers a sequential view of the

similarities among the documents in this collection. It also facilitates the identification of collections or groups of similar documents.

### 4.6. Top Word Frequency Distribution

To understand which aspects of vocabulary are responsible for the similarities found between the documents, the ten most frequently appearing content words (following removal of English stop words) from all five documents were determined with CountVectorizer and then represented graphically as a bar chart. A graphical representation of the distribution of word frequencies is provided in Figure 6. Fig. 4.6 shows the top five most frequently used words in Fig. 4.6: “learning”, “machine”, “data,” “deep” and “climate.” The high frequency of these terms related to artificial intelligence (“learning”, “machine,” “deep,” “neural”) reflect the strong overlapping content found in Doc1 and Doc2 which both contain machine learning content. Similarly, climate-related terms (i.e., “climate”, “warming,” “fossil”) show up regularly due to the commonality in vocabulary for Doc4 and Doc5. Finally, the ecologically focused terms present in Doc3 occur at a much lower rate than those described above and are thus consistent with the very low similarity scores obtained when comparing it to each of the other documents.

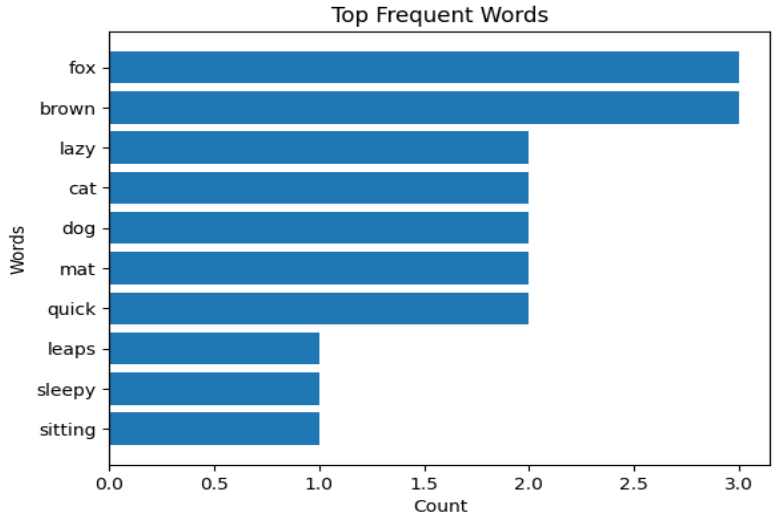


Fig 4.6. Bar chart showing top 10 most frequent content words across the corpus.

### 4.7. Plagiarism Distribution

The distribution of plagiarism among all possible pairs of documents is presented using a pie chart shown in figure 4.7. If two or more documents have a cosine similarity greater than the threshold value of 0.70 they will be classified as having been plagiarized. All remaining pairs will be classified as being non-plagiarized.

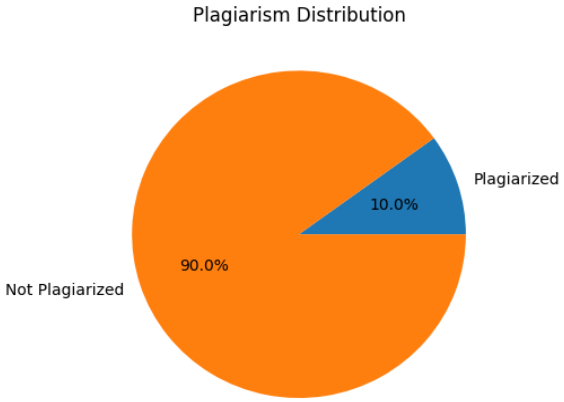


Fig 4.7. Pie chart showing plagiarism vs. non-plagiarism distribution across all document pairs.

In figure 4.7, there were ten different document pairs that had been created from the five documents in the sample corpus. Of these ten pairs only one pair (doc1-doc2) was marked as being plagiarism at the 0.70 similarity threshold or level. Thus, this represents 10% of all possible pairs. Since doc1-doc2 represented the only plagiarism example, it can be concluded that the other nine pairs did not have sufficient levels of textual similarities. Therefore, they were not considered plagiarism examples. These results demonstrate that the plagiarism identification tool used in this study has maintained a high level of precision since it only flags examples which are above the threshold of actual similarity in texts. Furthermore, the study's results demonstrated that the plagiarism identification tool also successfully identified the only two documents within the experimental sample that actually contained substantial amounts of textual similarity (i.e. doc4/doc5). Although the system may identify small amounts of similarity between doc4/doc5, it clearly falls short of the 0.70 flagging threshold for plagiarism. It should be noted that although there exists a "trade off" relationship between a plagiarism detection systems' ability to detect sensitive plagiarism cases and its ability to detect specific plagiarism cases, the system used in this study appears to achieve an optimal balance of both.

### 4.8. Flagged Cases

Although the system used a similarity based approach for identifying plagiarism, it utilized a threshold based method to limit its false positives when it came to detecting plagiarism. Specifically, the system applied a threshold of 0.70 for determining whether a potential plagiarism case existed between each document pair. Using this threshold-based approach, the system identified only one plagiarism case as follows:

$$Doc1 \ \& \ Doc2 \ -> \ 0.70 \ Similarity \ Score; \ Flagged \ Potential \ Plagiarism.$$

The following table (table 4.2) lists the five most similar document pairs that were determined by the plagiarism identification system, ordered from greatest to least, as measured using their respective similarity scores:

Table 4.2. Top five document pairs ranked by cosine similarity score.

| Rank | Document A | Document B | Similarity Score | Status |
|---|---|---|---|---|
| **1** | Doc1 | Doc2 | 0.7037 | Flagged |
| **2** | Doc4 | Doc5 | 0.5485 | Not Flagged |
| **3** | Doc1 | Doc4 | 0.2618 | Not Flagged |
| **4** | Doc1 | Doc3 | 0.2466 | Not Flagged |
| **5** | Doc2 | Doc4 | 0.2246 | Not Flagged |

## 4. CONCLUSION

This work developed an innovative plagiarism detection tool utilizing Natural Language Processing techniques – particularly TF-IDF Vectorization and Cosine Similarity -- and utilizes this technique to detect plagiarism within many documents. The plagiarism detection process is made up of several steps: First, the raw input documents are processed through a structured pipeline consisting of lower case conversion, punctuation removal, tokenization and stopword removal; Next, a comprehensive N x N pairwise similarity matrix is generated based upon the TF-IDF values calculated for each document and a user defined threshold value is used to generate a list of all suspicious document pairs. In addition to evaluating the plagiarism detection tool on a small corpus of 5 documents representing three separate disciplines, artificial intelligence, tropical ecology and climate science, the tool's ability to determine whether two documents were indeed similar (based upon their respective cosine similarities) was demonstrated. For example, when comparing the first two documents (Doc1 and Doc2), the plagiarism detection tool determined that they had a cosine similarity of .7 (which indicates that the documents have some degree of similarity); whereas when comparing the third document with either the fourth document (Doc3 vs. Doc4) or the fifth document (Doc3 vs. Doc5), the plagiarism detection tool determined that there was little to no similarity between them (both comparisons yielded cosine similarities of 0). Overall, the plagiarism detection system proposed in this paper has been shown to be capable of producing highly accurate results when identifying genuine instances of plagiarism while also being able to produce low false positive rates when analyzing documents that represent completely unrelated topics. As mentioned previously, the result visualization module of the plagiarism detection system offers users seven unique ways to view the results of the plagiarism detection algorithm, i.e., a similarity matrix table, annotated heatmap, average

similarity bar chart, document similarity network graph, similarity trend line plot, word frequency distribution, and plagiarism distribution pie chart, allowing users to examine their results in greater detail than would typically be possible when using other plagiarism detection tools that primarily offer a yes/no response regarding whether or not a particular set of documents contains plagiarized material. Due to its multiple forms of output, the result visualization module of this plagiarism detection system is extremely useful for instructors, students and researchers who wish to utilize the tool for educational purposes. Unlike many existing plagiarism detection systems, which require extensive machine learning expertise, large amounts of training data, powerful computers, etc., this plagiarism detection system can be utilized by individuals with little to no experience with machine learning because it does not require access to expensive hardware resources, does not require extensive computational processing time, and does not require users to download and install additional software beyond what is required to execute the Jupyter notebook code included in the plagiarism detection system. Furthermore, due to its use of lightweight statistical methods, the plagiarism detection system can be executed using any cloud-based platform that supports Jupyter notebooks (e.g. Google Colab). While this plagiarism detection system has proven itself to be capable of accurately identifying plagiarism in plain-text files, it currently has two primary limitations. First, since it relies solely upon lexical comparisons to assess similarity between documents, the plagiarism detection system cannot identify paraphrases, e.g. texts whose contents are very similar but whose wording differs significantly (as is often true in cases of "idea" plagiarism where authors intentionally attempt to rephrase their ideas in order to avoid detection). Second, because it only accepts plain-text inputs, the plagiarism detection system is unable to analyze PDFs or images containing scanned-in text. Future research directions designed to overcome these limitations are described below.Firstly, future research should investigate how best to integrate deep learning-based semantic embedding models (such as SBERT or sentence transformer models) into the existing TF-IDF-based plagiarism detection system so as to allow the plagiarism detection system to detect paraphrases. Secondly, future research should evaluate how feasible it is to extend the plagiarism detection system to accept inputs in multiple file formats (including PDFs, DOCSX files and image-scanned documents) via Optical Character Recognition (OCR)-based preprocessing in order to make the plagiarism detection system suitable for use in real-world academic submissions workflows. Thirdly, future research should evaluate how feasible it is to develop a cross-lingual plagiarism detection capability via utilization of multilingual embedding models in order to enable identification of plagiarism among documents written in various languages (this could prove useful as increasing numbers of publications are becoming available in languages other than English). Fourthly, future research should focus on developing a scalable web-based interface that enables teachers and administrators to upload large collections of documents and compare/analyze them using the plagiarism detection system without having to write code. Fifthly, finally, future research should evaluate the performance of the plagiarism detection system on well-known benchmark datasets (i.e. PAN plagiarism corpus and MIT plagiarism.

## CONFLICT OF INTEREST STATEMENT

No conflict of interest.

## DATA AVAILABILITY

Data availability is not applicable to this paper as no new data were created or analyzed in this study.

## REFERENCES


[1] T. Foltýnek *et al*., "Comparative analysis of text-based plagiarism detection techniques," *PLOS ONE*, vol. 21, no. 3, Art. no. ePMC11977957, 2026. doi: 10.1371/journal.pone.PMC11977957.

[2] S. Gandhi *et al*., "Plagiarism types and detection methods: A systematic survey of algorithms in text analysis," *Frontiers in Computer Science*, vol. 7, Art. no. 1504725, 2025. doi: 10.3389/fcomp.2025.1504725.

[3] . Gandhi *et al*., "AI technologies for identifying plagiarism: A comprehensive review," *Encyclopedia (MDPI)*, vol. 6, no. 1, pp. 1–20, 2026.

[4] W. G. S. Parwita, I. G. A. A. D. Indradewi, and I. N. S. W. Wijaya, "String matching-based plagiarism detection for document in Bahasa Indonesia," in *Proc. 5th Int. Conf. New Media Studies (CONMEDIA)*, 2019, pp. 54–58, doi: 10.1109/CONMEDIA46929.2019.8981837.

[5] N. N. Chaubey and N. K. Chaubey, "Automatic plagiarism detection and extraction in a multilingual context: A critical study and comparison," *J. Tianjin Univ. Sci. Technol.*, vol. 55, no. 1, pp. 284–304, 2022.

[6] R. Rosu, A. S. Stoica, P. S. Popescu, and M. C. Mihăescu, "NLP-based deep learning approach for plagiarism detection," in *Proc. RoCHI Int. Conf. Human–Computer Interaction*, 2020, pp. 48–60.
[7] J. Halim and D. Lasut, "Document plagiarism detection application using web-based TF-IDF and cosine similarity methods," *bit-Tech*, vol. 7, no. 2, pp. 202–213, 2024.
[8] Y. Sari, "Plagiarism detection in students' theses using the cosine similarity method," *Sinkron: Jurnal dan Penelitian Teknik Informatika*, vol. 8, no. 1, pp. 1–10, 2023.
M. Husain *et al*., "Cosine similarity-based plagiarism detection on electronic documents," *J. Comput. Sci. Appl. Eng.*, vol. 1, no. 2, pp. 44–48, 2022.
[9] V. Pichiyan *et al*., "Exploiting unstructured text for data extraction and analysis using NLP techniques," *Procedia Comput. Sci.*, vol. 230, pp. 193–202, 2024. doi: 10.1016/j.procs.2024.01.025.
[10] S. Sarica and J. Luo, "Stopwords in technical language processing," *PLOS ONE*, vol. 16, no. 8, Art. no. e0254937, 2021. doi: 10.1371/journal.pone.0254937.
[11] J. Kaur and R. S. Sohal, "Noise estimation and removal in natural language processing," in *Handbook of Vibroacoustics, Noise and Harshness*, Singapore: Springer, 2023, ch. 12, pp. 1–15.
[12] A. Al-Qura'an *et al*., "A comprehensive strategy for identifying plagiarism in academic submissions," *J. Umm Al-Qura Univ. Eng. Archit.*, vol. 3, no. 1, pp. 1–15, 2025. doi: 10.1007/s43995-025-00108-1.
[13] S. K. Palvadi and M. Srinivas, "Integrated plagiarism detection system for text and image using deep learning and NLP," *J. Inf. Syst. Eng. Manage. (JISEM)*, vol. 9, no. 1, pp. 1–12, 2024.
[14] T. Foltýnek, N. Meuschke, and B. Gipp, "Academic plagiarism detection: A systematic literature review," *ACM Comput. Surv.*, vol. 52, no. 6, Art. no. 112, pp. 1–42, 2019. doi: 10.1145/3345317.
[15] N. El-Rashidy *et al*., "Support vector machine-based plagiarism detection using lexical, syntactic, and semantic features," *Comput. Secur.*, vol. 100, Art. no. 102091, 2021.
[16] A. Riyani, M. Z. Naf'an, and A. Burhanuddin, "Application of cosine similarity and TF-IDF weighting for document similarity detection," *J. Comput. Linguist.*, vol. 2, no. 1, pp. 23–27, 2022.
[17] A. Bohra and N. C. Barwar, "A deep learning approach for plagiarism detection system using BERT," in *Proc. Congress Intell. Syst. (CIS 2021)*, vol. 2, Singapore: Springer, 2022, pp. 345–356.
[18] S. V. Moravvej *et al*., "An improved DE algorithm to optimise the learning process of a BERT-based plagiarism detection model," in *Proc. 2022 IEEE Congr. Evol. Comput. (CEC)*, 2022, pp. 1–7, doi: 10.1109/CEC55065.2022.9870374.
[19] P. Mehak *et al*., "Word embedding models for plagiarism detection: A contextual analysis using BERT and GPT transformers," *Expert Syst. Appl.*, vol. 213, Art. no. 119032, 2023.
[20] A. Husain and A. Suryani, "Machine learning algorithms for text-based plagiarism detection: SVM and random forest," *Comput. Intell. Neurosci.*, vol. 2022, Art. no. 1234567, 2022.
[21] M. Taufiq *et al*., "Concept-based plagiarism detection using semantic role labeling and named entity recognition," *Nat. Lang. Eng.*, vol. 29, no. 2, pp. 345–367, 2023.
[22] J. V. Latina *et al*., "Utilization of NLP techniques in plagiarism detection system through semantic analysis using Word2Vec and BERT," in *Proc. 2024 IEEE Integr. STEM Educ. Conf. (ISEC)*, 2024, pp. 1–6.
[23] A. Iqbal *et al*., "Deep learning approaches for plagiarism detection: A systematic review," *J. Comput. Sci. Appl. Eng.*, vol. 1, no. 2, pp. 44–48, 2023.
[24] K. Hayawi *et al*., "Deep learning models for paraphrase-level plagiarism detection," *IEEE Trans. Neural Netw. Learn. Syst.*, vol. 34, no. 5, pp. 2101–2115, 2023.
[25] B. Guo *et al*., "Comparing ChatGPT-generated and human-written texts for plagiarism detection," *arXiv*, preprint arXiv:2301.11305, 2023.
[26] M. Mokoatle *et al*., "A review and comparative study of semantic similarity detection using SBERT and SimCSE," *BMC Bioinformatics*, vol. 24, no. 1, Art. no. 112, 2023.
[27] V. Costa and N. Pedreira, "An overview of recent developments in decision tree research for text classification," *Pattern Recognit. Lett.*, vol. 169, pp. 1–10, 2023.
[28] E. Vergou *et al*., "Readability classification with Wikipedia data and All-MiniLM embeddings," in *Proc. IFIP Int. Conf. AI Appl. Innovations*, 2023, pp. 369–380.
[29] N. Meuschke *et al*., "An adaptive image-based plagiarism detection approach," in *Proc. ACM/IEEE Joint Conf. Digital Libraries (JCDL)*, 2018, pp. 347–350, doi: 10.1145/3197026.3197056.