International Journal of Engineering Technology and Management Sciences
Website: ijetms.in Issue: 3 Volume No.8May - June – 2024
DOI:10.46647/ijetms.2024.v08i03.072 ISSN: 2581-4621

# Text Similarity Verification

[1]Ms.Naheed Sultana, [2]G.Sahiti, [3]M.Vanshita Singh, [4]K.Tejaswi
[1]*Assistant Professor, Dept. of IT, Stanley College of Engineering and Technology for Women, India*
[2]*Student, Dept. of IT, Stanley College of Engineering and Technology for Women, India*
[3]*Student, Dept. of IT, Stanley College of Engineering and Technology for Women, India*
[4]*Student, Dept. of IT, Stanley College of Engineering and Technology for Women, India*

***Abstract--***In the rapidly evolving landscape of natural language processing (NLP), accurately measuring the similarity between texts is paramount. This project explores the implementation of text similarity verification using the Term Frequency-Inverse Document Frequency (TF-IDF) approach, a well-established technique in information retrieval and text mining. The primary objective of this study is to develop and evaluate a system that can effectively assess the similarity between textual documents by leveraging the TF-IDF methodology. The project encompasses the following key components: data pre-processing, TF-IDF vectorization, and similarity computation. Data pre-processing involves cleaning and preparing textual data to ensure high-quality input for analysis. The TF-IDF vectorization process transforms the preprocessed text into numerical vectors that reflect the importance of terms within the document corpus. This project not only reaffirms the utility of TF-IDF in text similarity verification but also provides a foundation for future research and development in enhancing text analysis techniques.
***Key Words:*** **Term Frequency, Inverse Document Frequency, Text Preprocessing, Stop Words, Cosine Similarity, Euclidian Distance, Jaccard Similarity, Manhattan Distance.**

## I. Introduction
In the digital age, the proliferation of text-based information across the internet, academic databases, and various digital platforms has created a pressing need for efficient and accurate text similarity verification. In an era dominated by vast amounts of textual data generated across diverse digital platforms, the ability to accurately measure and verify text similarity has become increasingly important. From academic institutions monitoring for plagiarism to search engines optimizing query results, text similarity verification is a critical component in numerous applications. Text similarity verification involves determining how closely related two pieces of text are, which can be applied in various domains such as plagiarism detection, document clustering, and recommendation systems. The significance of text similarity verification is evident in academia, where it is used to uphold the integrity of scholarly work by detecting instances of plagiarism. Search engines, enhance the relevance of retrieved information by understanding user queries in a more nuanced manner. The findings of this project are expected to contribute to the advancement of NLP applications and enhance the accuracy and reliability of text comparison tasks in diverse fields. Various applications of Text Similarity Verification include Information Retrieval, Content Recommendation Systems, Document Clustering and Categorization, Paraphrase and Duplicate Detection, and Legal Document Analysis.

### A.Problem Statement
To prepare a Text Similarity Verification that utilizes the Term Frequency-Inverse Document Frequency (TF-IDF) technique for generating the similarity score between the two input statements or between the input statement and the uploaded doc which includes the percentage of similarity, list of similar words, and graphical representation for it.

### B. Aim and Scope
This project aims to implement and evaluate various text similarity verification methods using the Term Frequency-Inverse Document Frequency (TF-IDF) representation combined with different

similarity metrics. This project seeks to determine the effectiveness of these combinations in identifying the similarity between text documents. The project has laid a solid foundation, and there are several directions in which it can be expanded and improved such as Enhanced Pre-processing, Synonym Handling; User Experience, Drag-and-Drop Uploads, Batch Processing; Scalability and Performance, Database Integration; Additional Features, and Detailed Reports.

## II. Existing System

In the past, text similarity verification was a manual process involving the comparison of documents to identify similarities. This method was very time-consuming, tedious, and subject to errors made by humans. Also, it was not effective when dealing with large amounts of text. As a result, it lacked efficiency and scalability. The Existing System divides text similarity into two main categories Text distance and Text representation but it lacks practical examples or case studies that demonstrate the application of these methods in real-world scenarios. It covers traditional methods like vector space models, which may have too many equations and calculations, which seems difficult for one with no mathematical background. It explores various algorithmic approaches including token-based, tree-based, and graph-based methods. Some existing software solutions offered basic text similarity verification capabilities, but they often suffered from limitations such as:

### A.Drawbacks

Detailed mathematical formulations and technical descriptions might be challenging for readers without a strong background in mathematics or NLP.
Require large corpora for training, computationally intensive, may not work well with limited data.
Depending on the availability and completeness of knowledge bases, may not scale well with large datasets, and computational complexity can be high.
Does not cover recent developments in deep learning extensively.
The fast-paced advancements in NLP may quickly outdate some findings and recommendations.

## III. Methodology

The machine works with number vectors and the data is in the form of text. It is necessary to convert text into a digital format which is done through a process called vectorization. One popular method for converting text into vectors is the TF-IDF method, which calculates the term frequency of a word in a document. Counting the occurrences of the word throughout the document is one of the most straightforward methods. TF-IDF measures how common or rare a word is within a set of documents. This is calculated by dividing the total number of documents by the number of documents containing the word and then calculating the logarithm. One effective way to measure the similarity between two documents is by using the Inverse Document Frequency (TF-IDF) technique along with cosine similarity.

**TF-IDF Vectorization:** TF-IDF (Term Frequency-Inverse Document Frequency) vectorization is a common technique used in natural language processing (NLP) and information retrieval to represent text data numerically. It's particularly useful for tasks like document classification, clustering, and similarity calculation.TF-IDF is calculated by multiplying two metrics:

**Term Frequency (TF):** The formula divides the number of times a term appears in the document by the total number of terms.

*TF*term=No. of times term appears in a document/Total no. of terms in the document

**Inverse Document Frequency (IDF):** IDF measures the importance of a term across the entire corpus (collection of documents). It's calculated as the logarithm of the ratio of the total number of documents to the number of documents containing the term.

IDF (t, corpus)=log (Total number of texts in corpus/Number of texts containing term t)

TF-IDF (t, text, corpus) = TF (t, text)*IDF (t, corpus)

By employing TF-IDF vectorization along with cosine similarity, Euclidian Distance, Jaccard Similarity, and Manhattan Distance the project effectively captures the similarity between the two query texts, providing a robust mechanism for comparing textual data and determining their similarity along with their graphical representation.

**A.Proposed System**

Text Similarity is the process of comparing a piece of text with another and finding the similarity between them. It’s basically about determining the degree of closeness of the text. Dealing with text, sentences or words brings us to the region of Natural Language Processing (NLP).

**How does Text Similarity work?**

To start with the text similarity task, firstly we need to convert the input text into a more machine-readable form by transforming the text into numerical vectors that are understood by the machine to calculate the similarity. To convert text into vector the TF-IDF algorithm is used.

**How to get ready for the Text Similarity task?**

**Text Pre-processing**

**Tokenization:** Tokenization is a crucial step in text processing, especially for tasks like text similarity verification. It involves breaking down a text into smaller components, such as words or phrases, which are called tokens. These tokens can then be analyzed to determine how similar two pieces of text are.

**Example:** The quick brown fox jumps over the lazy dog
[“The”, “quick”, “brown”, “fox”, “jumps”, “over”, “the”, “lazy”, “dog”]

**Stop Word Removal:** Stop words are commonly used words in a language that are often considered to have little lexical content or value in the context of text analysis. Examples in English include "the," "is," "in," "and," etc. Stop words can add noise to the data, making it harder to identify the significant terms that contribute to the actual meaning of the text so removing stop words helps in reducing the size of the text data, making computations faster and more efficient.

**Example:** The quick brown fox jumps over the lazy dog
[“quick”, “brown”, “fox”, “jumps”, “lazy”, “dog”]

**The Proposed System includes the following features**

***Automated Text Processing:*** Unlike the manual input required by previous systems, the proposed system automates text processing tasks such as lowercase conversion and punctuation removal. This advanced automation enhances the detection process, optimizing efficiency and empowering users to concentrate on result interpretation rather than data pre-processing.

***Advanced Algorithms:*** The proposed system leverages advanced algorithms such as TF-IDF vectorization along with cosine similarity, Euclidian Distance, Jaccard Similarity, and Manhattan Distance metrics for accurate and reliable text similarity along with graphical representation for the individual similarity score. These algorithms enable a deeper analysis of text similarity, resulting in more precise detection of plagiarized content, even in cases of paraphrasing or disguised plagiarism.

***Robust Error Handling:*** The proposed system incorporates robust error handling mechanisms to address exceptions that may occur during operation. Detailed error messages, error logs, and exception-handling routines provide users with actionable feedback and guidance, minimizing disruptions and maximizing productivity.

***Report:*** The system provides us with the similarity score showing the list of words that are similar in the provided text along with the graphical representation.

**B. Advantages**

Implementing a text similarity verification system using TF-IDF (Term Frequency-Inverse Document Frequency) along with various similarity and distance metrics offers several advantages. Each method has its unique strengths and can be used to analyze different aspects of text similarity

***1. Importance Weighting:*** TF-IDF assigns higher weights to more important terms within a document and is less frequent across the corpus. This helps in emphasizing significant words while downplaying common words like "the" or "and".

***2. Sensitive to Magnitude:*** Since it considers the magnitude of vector differences, it can be effective in identifying documents with similar term frequencies.
***3. Angle-Based Similarity:*** Cosine similarity measures the cosine of the angle between two vectors, making it useful for determining orientation rather than magnitude. This is particularly beneficial when the length of documents varies significantly.
***4. Simplicity:*** Jaccard Similarity provides a simple yet effective measure of similarity that is easy to interpret and compute.
***5. Efficiency:*** It's computationally less intensive and simpler to implement, making it suitable for large-scale text comparison tasks.

## C. System Architecture

The architecture of the project is organized into several layers, each serving a specific purpose in facilitating text comparison and similarity assessment. At the forefront is the User Interface (UI) Layer, responsible for presenting the graphical interface to users. This layer includes components such as labels, entry widgets for inputting query text, and buttons for uploading files and initiating similarity checks. The File Handling Layer manages file operations, enabling users to upload text files for comparison.
Next, the Text Pre-processing Layer prepares the query text and database text for comparison. This ensures uniformity and consistency in the comparison process. The Text Comparison Layer utilizes advanced algorithms like TF-IDF vectorization, cosine similarity, Jaccard similarity, Euclidian distance, and Manhattan distance calculation to determine the similarity between the two query texts or the text from the database. It transforms textual data into numerical vectors and computes similarity scores, providing accurate results along with the graphical representation of the individual similarity score.
Error handling is managed by the Error Handling Layer, which catches and handles exceptions gracefully, ensuring a smooth user experience. This layer utilizes message box prompts to communicate errors, such as file not found or invalid data entry, to the user effectively. It facilitates data flow and control between UI, file handling, text pre-processing, text comparison, and error handling layers, promoting modularity and maintainability of the system.

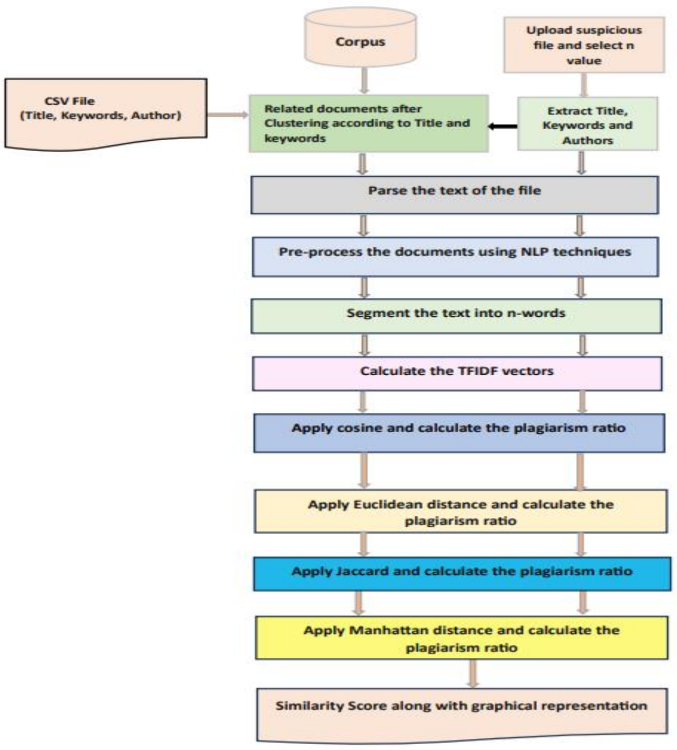

Corpus
Upload suspicious file and select n value
CSV File (Title, Keywords, Author)
Related documents after Clustering according to Title and keywords
Extract Title, Keywords and Authors
Parse the text of the file
Pre-process the documents using NLP techniques
Segment the text into n-words
Calculate the TFIDF vectors
Apply cosine and calculate the plagiarism ratio
Apply Euclidean distance and calculate the plagiarism ratio
Apply Jaccard and calculate the plagiarism ratio
Apply Manhattan distance and calculate the plagiarism ratio
Similarity Score along with graphical representation

**Fig: Flow chart of Text Similarity Verification**

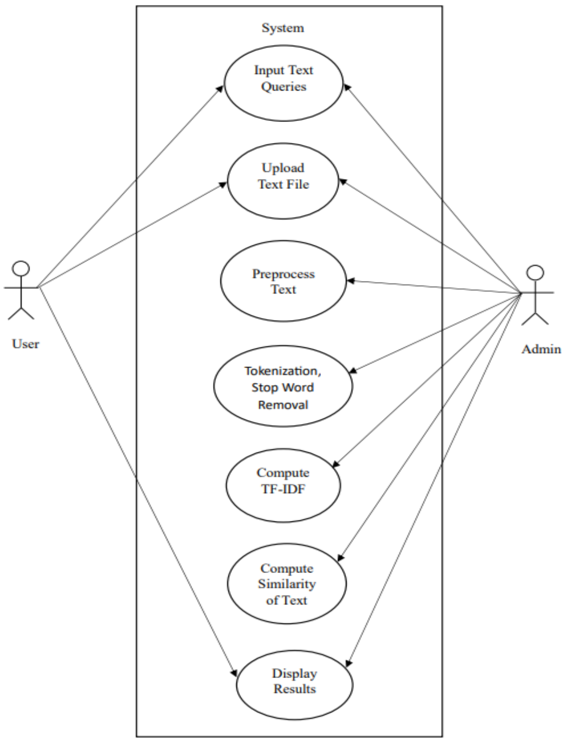


Fig: Use-Case Diagram

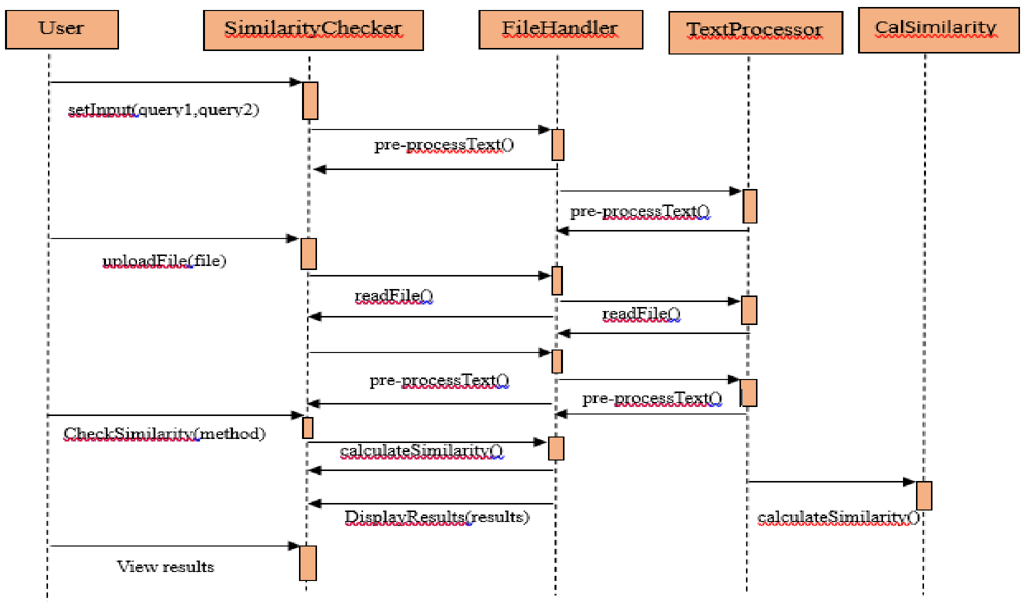


Fig: Sequence Diagram

## IV. Implementation

### 1. Text-preprocessing module:

- Pre-processing is a major step in the process. It is an important step to detect plagiarism. In this step, data is processed in an appropriate form which can be inputted for the detection process. The submitted docs are of different formats including lower & upper case letters. So to remove the sensitivity, all documents are converted into one format.
- The first step in plagiarism detection is pre-processing the text. This involves preparing the documents for comparison by removing irrelevant information such as punctuation, capitalization, and stop words. Pre-processing text helps to improve object detection accuracy by minimizing noise and simplifying the comparison process. Instead of getting distracted by surface-level differences,

the focus should be on identifying similarities in the core content of the documents. The techniques used in the text pre-processing module are Tokenization and Stop Word Removal

**2. TF-IDF Vectorization:**

- **Term Frequency (TF):** Calculate the frequency of each term in the document.
- **Inverse Document Frequency (IDF):** Calculate the inverse document frequency for each term across all documents.
- **TF-IDF Calculation:** Multiply the TF of each term by its IDF.

**3. Text Comparison Module:**

- Performs the similarity comparison between the query text and the database text.
- The text provides functions for calculating TF-IDF vectors, computing cosine similarity scores, and identifying matched words between texts.

**A.Algorithms**

**Cosine Similarity**

The cosine similarity algorithm is used to determine how similar two vectors are, irrespective of their size. Determines document similarity based on word content, regardless of order.

**How does cosine similarity work?**

Cosine similarity ranges from -1 to 1; a score of 1 indicates identical texts, 0 means no similarity, and -1 denotes completely dissimilar texts.

Given two vectors *A* and *B* representing the TF-IDF representations of two texts:

**Cosine similarity (A,B)=A.B/||A||*||B||**

**Euclidian Distance**

The Euclidean distance formula can calculate the straight-line distance between two vectors. This involves taking the square root of the sum of the squared differences between the corresponding elements of the two vectors. A smaller distance indicates greater similarity.

**How does it work?**

1. **Squared Differences:** Calculate the squared difference between each pair of corresponding elements in the vectors.
2. **Square root of the sum:** Take the square root of the sum of squared differences.

**euclidean_distance = Sqrt ($\sum n\ i=1(Ai-Bi\ )$^2)**

**Jaccard Similarity**

Measures the ratio of the intersection to the union of two sets. Higher Jaccard similarity indicates greater similarity.

**How does it work?**

1. **Intersection and Union:** Identify the common elements (Intersection) and the total unique elements (Union) in both sets of words.
2. **Calculation:**

**Jaccard_similarity = Size of Intersection / Size of Union**

**Or**

**Number of common words / Total number of Unique words**

**Manhattan Distance**

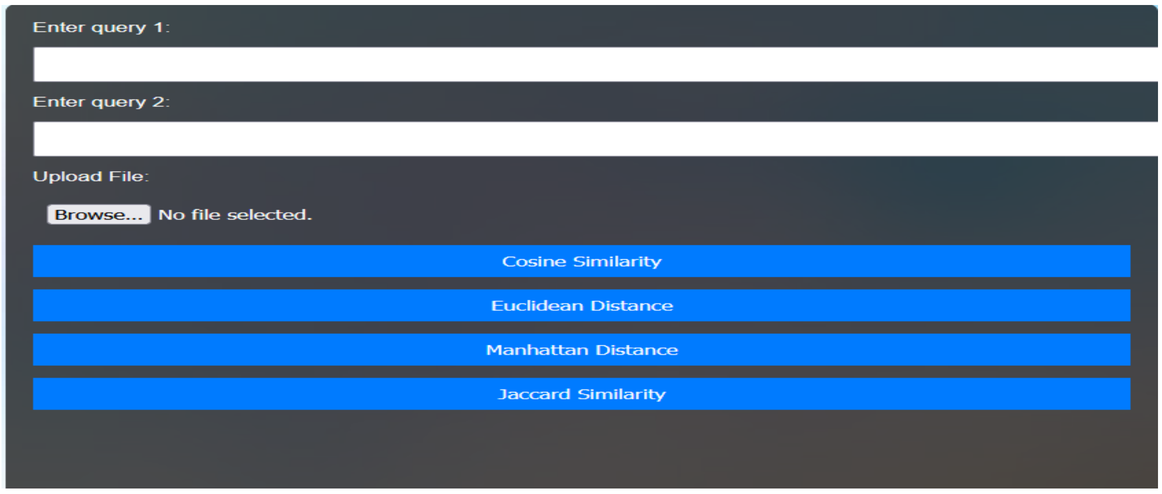
Manhattan distance can be employed to compare the similarity between two texts based on the differences in their word frequencies or other features. It is just one of many distance metrics used in text similarity tasks.

**How does it work?**

To use Manhattan distance for text similarity verification, you typically represent each text document as a vector where each dimension corresponds to a feature. These features could be word frequencies, TF-IDF scores, or any other relevant measure. Then, you compute the Manhattan distance between the vectors representing the two texts. Here's the formula for computing the Manhattan distance between two vectors a and b:

**Manhattan Distance (a,b)=$\sum n\ i=1|a\ i\text{-}b\ i\ |$**

## V. Result



## VI. Conclusion

The Similarity Checker project demonstrates a practical application of text similarity measures, including Cosine Similarity, Euclidean Distance, Jaccard Similarity, and Manhattan Distance along with a graphical representation for all the displayed similarity scores. The tool effectively pre-processes text, calculates similarity scores, and also provides a list of matching words between two queries or documents. The project showcases how text analysis and natural language processing techniques can be integrated into a web-based interface, making it accessible and user-friendly. Key achievements of the project include **Text Pre-processing** for effective removal of stop words and normalization of text. **Similarity Calculation** helps in the Implementation of four distinct similarity measures, providing users with multiple perspectives on text similarity. **File Handling** can handle plain text and DOCX files, enhancing usability. And lastly, the **User Interface is** a responsive and visually appealing Ul that simplifies user interactions and presents results.

## References

[1] Measurement of Text Similarity: A Survey by Jiapeng Wang and Yihong Dong published in the year 2020.
[2] A Survey of Text Similarity Approaches by Wael H. Gomaa and Aly A. Fahmy was published in the year 2013.
[3] A Survey on Text Similarity Measurement. Charu C. Aggarwal and Cheng Xiang Zhai was published in the year 2012.
[4] Text Similarity and Plagiarism Detection: A Survey. Vipul Gupta and Sanjeev Sofat was published in the year 2014.
[5] “Introduction to Information Retrieval” This book provides a comprehensive introduction to information retrieval, including detailed explanations of TF-IDF and its applications in text similarity by Manning, C. D., Raghavan, P., & Schütze, H. Cambridge University Press was published in the year 2008.
[6] Short Text Similarity Measurement Methods: A Review by Dimas Wibisono Prakoso, Asas Abdi, and Chintan Amrit published in the year 2021
[7] Jones, K. S. A statistical interpretation of term specificity and its application in retrieval. Journal of Documentation, 28(1), 11-21. The foundational paper that introduces the concept of inverse document frequency (IDF).
[8] Ramos, J. Using TF-IDF to determine word relevance in document queries. Proceedings of the First Instructional Conference on Machine Learning. This paper explains the application of TF-IDF in determining word relevance within document queries, making it a practical guide.

[9] Sentence Similarity Based on Semantic Nets and Corpus Statistics by R. Mihalcea published in the year 2020.
[10] Online Plagiarism Detection and Result Evaluation using Data Mining and NLP by Aparna, Anju, John, and Divya published in the year 2018
[11] Sebastiani, F. Machine learning in automated text categorization. ACM Computing Surveys, 34(1), 1-47. This survey paper discusses various machine learning methods for text categorization, with references to the use of TF-IDF in these methods.