Received: 4 September 2025 | Revised: 8 December 2025 | Accepted: 13 July 2026 | Published online: 14 August 2026

RESEARCH ARTICLE

Journal of Data Science and Intelligent Systems
2026, Vol. 00(00) 1–7
DOI: 10.47852/bonviewJDSIS62027582

BON VIEW PUBLISHING

A Comparative Analysis of Two Algorithms for TF-IDF-Based Document Similarity

Glory Tendaishe Muindisi[1,*], Yusra Nuri[1] and Edip Senyurek[1]

[1] *Computer Engineering, Vistula University, Poland*

Abstract: By using Term Frequency–Inverse Document Frequency, this study set out to analyze how different algorithms carry out the task of identifying research articles that are considered to be of similar subjects. Two algorithms were implemented in our study, one of which was a Python algorithm based on cosine similarity and the other in C# using standard statistical measures. The goal was to examine how the ranking of documents to the most similar target article can be influenced by factors such as preprocessing, similarity techniques, and programming languages. The dataset contains a total of 50 abstracts, with two versions. One with common words removed and one left unchanged. The 50th abstract was selected as the target document, and since it included 29 references, we evaluated the top 29 results produced by each algorithm. The Python implementation relied on cosine similarity only, whereas the C# implementation used mean, standard deviation, Z-score, and weighted similarity. Performance was assessed using Spearman's rank correlation and hit ratio. The results showed that removing common words generally improved performance. The best result (75.86%) came from the C# algorithm using statistical measures with common words included. Overall, the findings highlight that algorithm choice and preprocessing decisions significantly affect document similarity outcomes.

Keywords: cosine similarity, TF-IDF, Spearman's rank correlation, text analysis

1. Introduction

To come up with certain ideas or reach a certain conclusion on different topics or to gather new information, verify it, and accept it as knowledge, people in various fields do their own research.

The sources for these research papers can vary depending on the needs of different people, but the common thing is that accessing the materials in this digital era has been much easier than ever. But the real challenge lies in finding reliable, relevant information on a specific subject.

Traditional keyword-based search methods often fall short, especially when similar ideas are expressed using different words. That's where the tools of text similarity come in handy, giving a better option to identify related content in a short amount of time. The main advantage of these tools is that they focus on the detailed meanings of the words rather than relying on superficial matches.

The classic approach for highlighting important terms in a collection of documents is Term Frequency–Inverse Document Frequency (TF-IDF) [1]. TF-IDF assigns numerical weights to each term based on two metrics: term frequency (TF), how often a word appears in a document, and inverse document frequency (IDF), which measures how much less common the word is across all documents.

TF-IDF enables us to pick out the more important words in selected documents and also compare their weighted value [2]. Consequently, TF-IDF puts more value on words that distinguish a document in the process reducing the importance of common words. By comparing the similarity of TF-IDF vectors, we can observe how related and/or similar two documents are. Our study also takes into consideration cosine similarity as TF-IDF alone falls short as it does not account for the semantic meaning of words, which means it doesn't take the sentence structure into consideration. So cosine similarity will assist us in measuring similarities of documents not only in word frequency but also semantically.

Our first algorithm, implemented in Python, was paired with cosine similarity to compare the document similarity of a 50-document dataset. Each document was transformed into a numerical vector using TF-IDF and then utilized cosine similarity to measure the degree of similarity between these vectors. Instead of quantifying the shared words, this method calculates the cosine of the angle between the vectors. A smaller angle indicates a higher level of similarity, implying the documents contain similar content. This approach is particularly effective in text analysis because it has a negligible impact on document length and focuses on the distribution of terms rather than their absolute frequency. Resultantly, it allows for a more balanced and context-aware comparison of documents, making it well-suited for identifying relevant research articles within a large dataset [3].

Our second algorithm, which we implemented in C#, relies on a set of statistical measures derived from the TF-IDF representation of each document; we further processed these values by computing statistical indicators such as mean, standard deviation, and Z-score, which help describe the distribution and variation

*Corresponding author: Glory Tendaishe Muindisi, Computer Engineering, Vistula University, Poland. Email: gmuindi1@stu.vistula.edu.pl

of term importance within each document. These measures were then combined to produce a weighted similarity score, which helped in deciding how similar a document was to the target abstract. This helped us to compare not only how accurate the rankings were but also how similar the results from the two implementations were. Also included in the study are the effects of the presence and absence of common words in the dataset on both algorithms.

The experimental setup was based on the work of Nuri and Senyurek [4]. A dataset of 50 documents is used, with the references cited in the 50th document serving as a ground for evaluation. We focused on the top 29 ranked results because the target document contained 29 references, which gave us a practical way to measure how effectively each implementation identified relevant research articles from the dataset.

Although our implementations were in Python and C#, the goal of this study is not to compare the programming languages but instead to compare two algorithms, namely, cosine similarity applied in Python and normalization, standard deviation, Z-scores, and weighted similarity employed in C#. So, the aim is to evaluate how these two approaches affect the results of similarity rankings when using the same datasets.

This study is beneficial to the scientific area by highlighting how implementation variation can affect document similarity results and by offering practical insights for researchers developing text-similarity tools.

2. Literature Review

TF-IDF calculates the relevance of words by multiplying two metrics: TF and IDF. Chen and Lin [1] expand on this approach to replace the TF with topic semantic relevance. Additionally, their model incorporates instruction-tuned large language models (IT-LLMs) to extract contextually rich topics from documents at hand. This enables a more effective way of retrieving information from documents since it is context-based. On the other hand, Dessi et al. [5] focused on determining patients' issues related to health in clinical texts by comparing TF-IDF with word embedding techniques.

To improve the accuracy of text classification, Liang and Niu [6] introduced a framework that integrates Bidirectional Long Short-Term Memory (LSTM) and Text Convolutional Neural Network (Text-CNN) architectures. This approach combined Word2Vecs with a modified TF-IDF formula so that both sequential and context-related features were effectively captured. Similarly, Alammary [7] tackled the classification of Arabic assessment questions by designing an enhanced TF-IDF model aligned to Bloom's taxonomy. In comparison to the results from TF-IDF and Term Frequency–Part of Speech Inverse Document Frequency (TFPOS-IDF), the proposed model showed better improvements statistically, such as in precision, recall, and accuracy.

Naïve Bayes Classifiers with TF-IDF and count vectorizer representations were used to conduct sentiment analysis on two movie review datasets [8]. As a result, TF-IDF showed a slightly better performance. Similarly, Du et al. [9] integrated TF-IDF with Bidirectional Encoder Representations from Transformers (BERT), Maximal Marginal Relevance, TextRank, and Latent Dirichlet Allocation (LDA) to improve and better classify topics related to flash floods across different social media platforms.

In their study, Iwendi et al. [10] proposed a TF-IDF algorithm with the Temporal Louvain approach to enhance document rough classification and text summarization efficiency. For the identification of suicidal intentions in clinical notes, Kareem and Obaid [11] developed a hybrid model, STFS_TF-IDF_HDSM, to enhance semantic analysis by employing semantic and textual feature scaling with sentiment analysis based on TF-IDF, along with the integration of a hybrid deep sequential model. As a result, this model managed to effectively predict suicidal sentiments by incorporating unidirectional and bidirectional techniques, achieving an accuracy of 98.46%.

To detect malicious activities, Kim and Kim [12] applied both TF-IDF and a sliding window algorithm, preprocessing the input data from the application programming interface. This method addressed one of the challenges in dynamic analysis by unifying the size of the data, which led to improved model performance. On the other hand, Liu et al. [13] dealt with text sentiment analysis, designing a weight distribution technique by integrating text sentiment analysis and a rule-based sentiment dictionary with TF-IDF. Their study shows that the precision of TF-IDF can be effectively increased when integrated with domain-specific text preprocessing methods.

While Wang [14] proposed a development of an automatic question-answering model to improve the efficiency of online education and reduce time consumption using TF-IDF and an unsupervised particle algorithm, resulting in high precision and reduced response time, Mutsaddi and Choudhary [15] applied TF-IDF with BERT to improve the accuracy and precision of plagiarism detection.

By developing an application, Xiang [16] addressed one limitation of TF-IDF in classifying literary texts, such as its difficulty in handling uneven data distributions when used with Random Forest Classifiers. Similarly, Mohammed and Omar [17] presented an exam question classifier model to classify exam questions into several areas, based on Bloom's taxonomy. But because their model focuses on a specific domain, it is not well-suited for classifying questions across multiple domains.

Using a content-based filtering technique in combination with TF-IDF, Raharjo and Arifin [18] developed a system for podcast recommendations in the education genre, with Spotify being the source. They managed to improve the user experience of podcast recommendations on Spotify by tackling the challenges of finding relevant podcasts. On the other hand, Saputro et al. [19] focused on plagiarism detection in the final-year students' research papers using TF-IDF and the Rabin–Karp algorithm.

A recommendation system was designed by Zhang [20] using two text vectorization algorithms, namely, TF-IDF and Word2Vec, focusing on comparing their efficiency and generalization ability of these algorithms in order to help enterprises choose better text representation methods for food reviews. In contrast, Senyurek and Kevric [21] focused on evaluating the performance of 11 different binary similarity metrics within the serendipity-driven framework. Among them, the one that got the best balanced result was the Yule similarity metrics when it comes to efficiency and the quality of recommendation, while the other metrics like Hamann and Dice gave poor quality results in capturing varied recommendations.

To design a wearable Internet of Things (IoT) product that better meets users' needs, Sul and Cho [22] combined information from user reviews using analysis of TF-IDF from interviews, and Raut et al. [23] used the same approach, applying TF-IDF and machine learning, to categorize opinions of students about online classes and, as a result, managed to find Random Forest yielding the most reliable results.

A similarity-based approach for improving text classification was proposed by Zagatti et al. [24] by examining the relationship

between text vectorization techniques and classification. The study explores how, by using cosine similarity, one can compare vectorization methods such as TF-IDF, Bag of Words, and Word2Vec to help identify the most effective approach. The authors found that analyzing the performance of classification with average similarity often results in higher accuracy and F1-scores. This method uses less computational power to reduce cost while still giving results similar to other methods, making it a good choice for large-scale text processing tasks. Hossain et al. [25] also created a way to check how consistent Jupyter notebooks are by comparing outputs from repeated runs of the same project. Their method looked at how similar the code outputs are each time by testing it on 100 notebooks, instead of only focusing on the code or whether it runs without errors. They then found out that different similarity measures can highlight changes in outputs more effectively. Their results also showed that reproducibility should not just be a simple yes or no, but rather something that can change depending on how similar the outputs are when running the codes.

A bilingual plagiarism detection system was proposed by Jayasundara and Tissera [26] to address the challenges when using mixed languages; in this case, Sinhala and English are used together. The system combines TF-IDF with cosine similarity to detect plagiarism. The system was built using a Flask backend, React frontend, and MongoDB, which resulted in a high accuracy of 85% and was efficient in processing medium-length documents. On the other hand, the study by Birthriya et al. [27] focused on a different application of TF-IDF within the field of cybersecurity, specifically for detecting smishing messages. Instead of using only similarity measures, they combined TF-IDF with fuzzy logic-based linguistic features and machine learning classifiers to improve the accuracy of message classification.

In this study, we compare two different algorithms for measuring document similarity. One is based on cosine similarity, while the other uses a normalized statistical measure. By comparing their results, we aim to find out which method performs better at identifying relevant documents. The findings of this study may help researchers improve text-similarity systems and make them more accurate, efficient, and reliable for different applications.

Table 1 summarizes the datasets, programming languages, and algorithms used across the reviewed literature to better highlight the applications of TF-IDF integrated with various tools and techniques used.

3. Research Methodology

This study ranks the textual similarity of 49 documents to a 50^{th}, which is the target article [4], and compares if different algorithms can pick out the 29 references of the 50^{th} document by picking the top 29 most similar documents. The dataset utilized

Table 1
Datasets, programming languages, and algorithms used by different researchers

Authors	Dataset	Algorithms	Prog. Lang.
Kareem and Obaid [11]	Suicidal Tweet Detection Dataset (STDD)	TF-IDF, hybrid deep sequential model (Bi-LSTM, GRU, attention)	Python
Mutsaddi and Choudhary [15]	Kaggle–MIT Plagiarism Detection Dataset	TF-IDF, Random Forest, Decision Tree	Python
Danyal et al. [8]	IMDb Movie Reviews and Rotten Tomatoes Reviews	TF-IDF and count vectorizer	Python
Senyurek and Kevric [21]	Serendipity-2018	11 similarity metrics	MATLAB
Zhang [20]	Food-Related Reviews	TF_IDF, Collaborative Filtering, Deep Learning (DL)	Python
Du et al. [9]	Food-Related Sina Weibo Microblog Dataset	TF-IDF, LDA	Python
Kim and Kim [12]	PE Malware Machine Learning	TF-IDF, sliding windows, LSTM	Python
Mohammed and Omar [17]	Open-Ended Exam Questions	TF-IDF	Python
Sul and Cho [22]	Amazon Smart IoT Product Reviews	TF-IDF	Python
Alammary [7]	Arabic Assessment Questions	Improved TF-IDF (Bloom's taxonomy adaptation)	Python
Raharjo and Arifin [18]	Spotify Educational Podcast	TF-IDF, chi-square, Support Vector Machine (SVM)	Python
Iwendi et al. [10]	Multi-Source Streaming and Text Corpus	TF-IDF and Temporal Louvain	Python/Java
Liu et al. [13]	Cries in the Drizzle Corpus	TF-IDF and sentiment dictionary	Python
Liang and Niu [6]	Sohu News	TF-IDF, LSTM, Text-CNN, Word2Vec	Python
Wang [14]	English Q&A University Dataset	TF-IDF, Word Segmentation Algorithm	Python
Dessi et al. [5]	N2c2 Obesity Dataset	TF-IDF, Word2Vec, SVM, CNN	Python
Chen and Lin [1]	Pseudo-Document Corpus for Entity Saliency	Sem-TF-IDF and IT-LLMs	Python
Raut et al. [23]	Hate Speech Tweet	TF-IDF, Naïve Bayes, and SVM	Python
Xiang [16]	University Literary Text Dataset	TF-IDF, Random Forest	Python

two sets of 50 articles, one with common words and the other without, with the 50th article abstract for similarity ranking.

Before applying the similarity algorithms, all documents were processed for uniformity and to reduce noise in the textual data. First, all text was normalized by converting it to uppercase and removing punctuation marks, such as commas, periods, hyphens, etc., as they don't have any contribution to semantic meaning computations.

Two versions of the dataset were prepared to evaluate the impact of common words. In the first dataset, all words were kept including common words like "the", "an", "is". In the second dataset, however, common words were removed by utilizing the NLTK English stop word list. No stemming or lemmatization techniques were applied so that words remained in their original forms.

To pick out the most similar articles utilized, TF-IDF was used to transform the textual data into a matrix of numerical features. This method evaluates the importance of a word in a document in relation to all documents in the dataset. The formulas are as follows:

TF measures how frequently a term appears in a document, as seen in Equation (1):

$$TF = \frac{\text{\# of term frequency}}{\text{total \# of terms in the document}} \tag{1}$$

Equation (2) shows IDF measures the relevance of a word across a set of documents. Both algorithms utilized TF-IDF.

$$IDF = \log\left(\frac{\text{\# of documents}}{\text{\# of documents with the term}}\right) \tag{2}$$

To measure the similarity between the TF-IDF vectors in the first algorithm, Python and cosine similarity were used. Cosine similarity measures the similarity between two documents by calculating the cosine of the angle between their respective TF-IDF vectors as shown in Equation (3). This value range is between 0 (no similarity) and 1 (identical).

$$\text{Cosine Similarity} = \cos(\theta) = \frac{\text{A.B}}{||\text{A}||\times||\text{B}||} \tag{3}$$

Where:

A and B are the TF-IDF vectors of the two documents.

A.B is the dot product of the TF-IDF vectors.

A and B are the magnitudes (lengths) of the two vectors.

While the second algorithm does not utilize cosine similarity to compare the TF-IDF of documents, it does so by analyzing the statistical distribution of the TF-IDF of the 49 documents to the 50th document by using mean, standard deviation, and Z-scores in C#.

To provide a clearer understanding of the differences between the two proposed approaches, Table 2 summarizes the key characteristics and distinctions between Algorithm 1 and Algorithm 2.

Mean, also known as average, is one of the measures of central tendency that is calculated by adding the total TF-IDF values in each document and dividing them by the total number of unique words. By calculating the mean, one can understand the typical pattern of the data, and Equation (4) shows how to calculate the mean:

$$\text{Mean} = \frac{\text{Sum of the TF} - \text{IDF values in a dataset}}{\text{Total number of unique words}} \tag{4}$$

Standard deviation is the measure of variation in a dataset. It tells how far the values deviate from the mean, and it is calculated as shown in Equation (5):

$$\text{Standard deviation} = \sqrt{\frac{\sum (x-\overline{x})^2}{n-1}} \tag{5}$$

Where:

X is each TF-IDF value.

$\overline{x}$ is the calculated mean.

$n-1$ is the total number of unique terms in the document.

Normalization means scaling data in numeric variables within the range of 0 and 1. It helps in comparing documents more accurately by using the mean values to adjust TF-IDF values, which improves the filtering process. It is calculated by the formula:

$$\text{Normalization} = x_i - \overline{x} \tag{6}$$

Where:

x_i is the set of TF-IDF values.

$\overline{x}$ is the mean value.

Z-score is a measure of how many standard deviations above or below the mean a data point is. It is calculated using the equation shown below:

$$Z - Score = \frac{N_i}{STD_i} \tag{7}$$

Where:

N_i is the normalized value.

STD_i is the calculated standard deviation.

Finally, weighted value is the summation of the product of $Z - score_i \times Z - score_{50}$ divided by the product of. It is calculated as seen in Equation (8):

$$W_{i,50} = \frac{\sum Z\text{-}score_i \times Z\text{-}score_{50}}{STD_i \times STD_{50}} \tag{8}$$

Where:

$Z\text{-}score_i$ is the Z-score of the ith document from the relevant 49 articles.

$Z - score_{50}$ is the Z-score of the 50th document.

Table 2
Comparison of Algorithm 1 and Algorithm 2

Feature	Algorithm 1	Algorithm 2
Programming language	Python	C#
Method	Cosine similarity	Statistical analysis
Calculations	Dot product and vector magnitude	Mean, STD, and Z-score
Normalization	Not explicitly applied	Applied before Z-score calculation

STD_i is the STD of the i^{th} document from the 49 articles.

STD_{50} is the STD of the 50^{th} document, which is the target document and the abstract of this study.

To compare the similarity ranking result between both algorithms and their respective datasets, Spearman correlation was used. As seen in Equation (9), Spearman correlation assesses the strength and direction of a monotonic relationship between two ranked variables.

$$SR = 1 - \frac{6\sum d_i^2}{n(n^2-1)} \tag{9}$$

Where:

SR = Spearman's rank correlation coefficient

d_i = the difference between the ranks of each pair

n = the number of data points

The Spearman similarity rankings between both algorithms and the different datasets, including and excluding common words, are illustrated in Figure 1, as shown below. It aims to visualize the comparison in the ranking of the 49 abstract documents to see which methods were adjacent in ranking. The most similar rankings were between Python with common words and Python without common words, with 69.73%. The letters in the figure represent the rankings as follows:

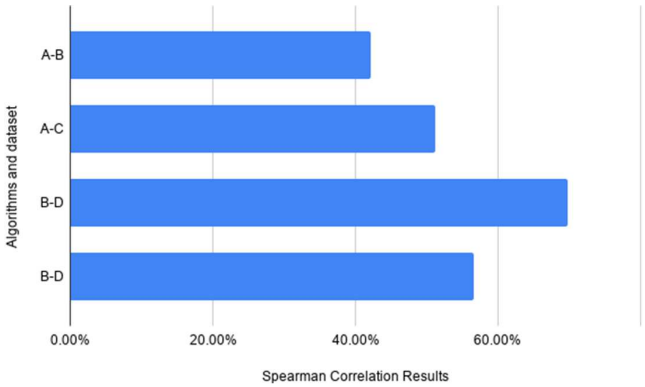
A: C# with common words

B: Python with common words

C: C# without common words

D: Python without common words

Figure 1
Spearman's correlation results comparing Algorithm 1 and Algorithm 2 with and without common words



A-B Algorithm 2 with common words and Algorithm 1 with common words

A-C Algorithm 2 without common words and Algorithm 2 common words

B-D Algorithm 1 with common words and Algorithm 1 with out common words

C-D Algorithm 2 common words Algorithm 1without common words

Algorithm 1—implemented and cosine similarity algorithm

Algorithm 2—C# and statistical measures

On the other hand, Table 3 shows the ranking results between Algorithm 1 and Algorithm 2, including and excluding common words.

To compare the 29 article references of the 50^{th} document to the top 29 articles out of the ranked 49 articles, the hit ratio was utilized. The study was set to find how many of the references showed up in the top 29 articles of each of the 4 rankings, which are Algorithm 1 with common words, Algorithm 1 without common words, Algorithm 2 with common words, and Algorithm 2 without common words. Equation (10) shows the hit ratio, and details can be seen in Figure 2, where E is the list containing the list of the 50th articles with 29 references.

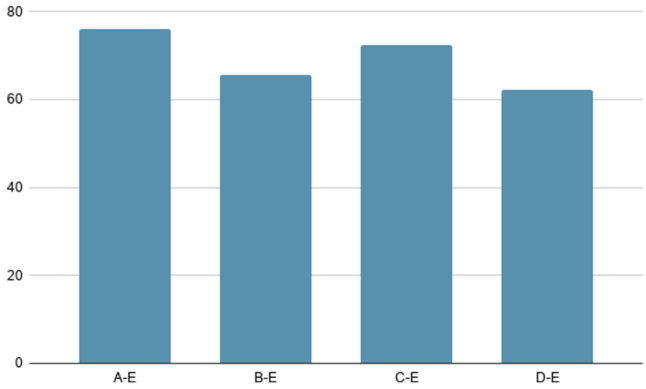
Table 3
Ranking results between Algorithm 1 and Algorithm 2, including and excluding common words

Algorithm	Common words	Rankings	Best
1	Yes	65.51%	3
2	Yes	75.86%	1
1	No	62.06%	4
2	No	72.41%	2

Algorithm 1- implemented and cosine similarity algorithm
Algorithm 2 – C# and statistical measures

$$Hit\ ratio = \frac{Number\ of\ hits}{Total\ number\ of\ documents} \tag{10}$$

Figure 2
Hit ratio comparison results



Rankings:

A-E is Algorithm 2 with the common words hit ratio with E

B-E is Algorithm 1 with a common words hit ratio with E

C-E is Algorithm 2 without common words, hit ratio with E

D-E is Algorithm 1 without common words hit ratio with E

Algorithm 1—implemented and cosine similarity algorithm

Algorithm 2—C# and statistical measures

E- the list of the 29 references of the 50^{th} document

4. Conclusion

In conclusion, the purpose of this paper was to observe the performance of two algorithms in picking textual similarities across selected documents in relation to a particular document, with one algorithm utilizing C# and statistical methods, and the second one implementing Python. We then compare which method of the two was more efficient at picking out the relevant references of the main document in a fixed dataset. In both algorithmic implementations, we used two datasets: one with common words and the other without. We employed TF-IDF in both algorithms; they then differed when calculating similarity ranking

between the 49 documents to the 50th document, with the first algorithm, which utilized C# and statistical formulas, namely, normalization, Z-score, and weighted balance, and the second algorithm, which used Python, made use of cosine similarity. For both algorithms and the respective datasets, we then observed how they ranked the 49 documents' similarity to the 50th. The results were tabulated in Table 2.

Lastly, we used the hit ratio to detect which implementation with the respective dataset had the most of the 29 article references of the 50th article in its top 29. The best-case scenario would have all 29 article references mentioned in the 50th document in its similarity ranking results list to the 50th document, and the worst case would have none. We then compared the hit ratio results across all implementations, which resulted in Algorithm 2 with common words dataset having the highest rate of identifying the references of the abstract document of 75.86%, followed by Algorithm 2 utilizing without common words that had 72.41%. Algorithm 2, which utilized statistical implementations, outperformed Algorithm 1 regardless of the dataset used. Algorithm 1, utilizing common words, obtained a hit ratio of 65.51%, and Algorithm 1 without common words resulted in the lowest hit ratio score of 62.06%. The dataset with common words in both algorithms outperformed the datasets without common words, with the difference between Algorithm 2 utilizing the dataset with common words and without being 3.45%, and the difference between the second algorithm and both datasets was 3.45%, which was the same.

One of the possible reasons Algorithm 2, which used C# and statistical methods, performed better, is that it not only compares TF-IDF values directly but also involves standard deviation from the average. This allows the program to notice even small differences that cosine similarity might smooth out. However, as only 50 abstracts were used in this study, the results are interpreted as indicative rather than conclusive. For future similarity ranking systems, this suggests that combining TF-IDF with statistical distribution analysis may be a good direction, especially for datasets where documents share the same vocabulary.

Ethical Statement

This study does not contain any studies with human or animal subjects performed by any of the authors.

Conflicts of Interest

The authors declare that they have no conflicts of interest to this work.

Data Availability Statement

The data that support the findings of this study are openly available in figshare at https://figshare.com/articles/dataset/Article_Abstracts/32834597?file=66102518.

Author Contribution Statement

Glory Tendaishe Muindisi: Methodology, Software, Validation, Formal analysis, Investigation, Writing – review & editing, Visualization. **Yusra Nuri:** Methodology, Software, Validation, Formal analysis, Investigation, Resources, Data curation, Writing – original draft, Writing – review & editing, Visualization. **Edip Senyurek:** Conceptualization, Resources, Data curation, Supervision, Project administration.

References

[1] Chen, W., & Lin, B. (2024). *Sem-TF-IDF: A simple semantic approach to generalize TF-IDF by employing instruction tuned large language models*. Technical Disclosure Commons. https://www.tdcommons.org/dpubs_series/6675/

[2] Nuri, Y., & Senyurek, E. (2024). Filtering articles based on their abstracts using TF-IDF. *International Journal of Advances in Engineering and Management*, *6*(8), 364–368.

[3] Widianto, A., Pebriyanto, E., Fitriyanti, F., & Marna, M. (2024). Document similarity using term frequency-inverse document frequency representation and cosine similarity. *Journal of Dinda: Data Science, Information Technology, and Data Analytics*, *4*(2), 149–153. https://doi.org/10.20895/dinda.v4i2.1589

[4] Nuri, Y., & Senyurek, E. (2025). Research abstracts similarity implementation by using TF-IDF algorithm. *IOSR Journal of Computer Engineering*, *27*(1), 4–10.

[5] Dessi, D., Helaoui, R., Kumar, V., Reforgiato Recupero, D., & Riboni, D. (2020). TF-IDF vs word embeddings for morbidity identification in clinical notes: An initial study. In *Proceedings of the First Workshop on Smart Personal Health Interfaces*, 1–12.

[6] Liang, M., & Niu, T. (2022). Research on text classification techniques based on improved TF-IDF algorithm and LSTM inputs. *Procedia Computer Science*, *208*, 460–470. https://doi.org/10.1016/j.procs.2022.10.064

[7] Alammary, A. S. (2021). Arabic questions classification using modified TF-IDF. *IEEE Access*, *9*, 95109–95122. https://doi.org/10.1109/ACCESS.2021.3094115

[8] Danyal, M. M., Khan, S. S., Khan, M., Ullah, S., Ghaffar, M. B., & Khan, W. (2024). Sentiment analysis of movie reviews based on NB approaches using TF–IDF and count vectorizer. *Social Network Analysis and Mining*, *14*(1), 87. https://doi.org/10.1007/s13278-024-01250-9

[9] Du, W., Ge, C., Yao, S., Chen, N., & Xu, L. (2023). Applicability analysis and ensemble application of BERT with TF-IDF, TextRank, MMR, and LDA for topic classification based on flood-related VGI. *ISPRS International Journal of Geo-Information*, *12*(6), 240. https://doi.org/10.3390/ijgi12060240

[10] Iwendi, C., Ponnan, S., Munirathinam, R., Srinivasan, K., & Chang, C.-Y. (2019). An efficient and unique TF/IDF algorithmic model-based data analysis for handling applications with big data streaming. *Electronics*, *8*(11), 1331. https://doi.org/10.3390/electronics8111331

[11] Kareem, F. R., & Obaid, A. J. (2025). STFS_TF-IDF _HDSM: Semantic and textual feature scaling using TF-IDF based sentiment analysis with Hybrid Deep Sequential Model. *International Journal of Information Technology*, *17*(9), 5751–5757. https://doi.org/10.1007/s41870-024-02330-x

[12] Kim, M., & Kim, H. (2024). A dynamic analysis data preprocessing technique for malicious code detection with TF-IDF and sliding windows. *Electronics*, *13*(5), 963. https://doi.org/10.3390/electronics13050963

[13] Liu, H., Chen, X., & Liu, X. (2022). A study of the application of weight distributing method combining sentiment dictionary and TF-IDF for text sentiment analysis. *IEEE Access*, *10*, 32280–32289. https://doi.org/10.1109/ACCESS.2022.3160172

[14] Wang, H. (2024). Automatic question-answering modeling in English by integrating TF-IDF and segmentation algorithms.

Systems and Soft Computing, *6*, 200087. https://doi.org/10.1016/j.sasc.2024.200087

[15] Mutsaddi, A., & Choudhary, A. P. (2025). Enhancing plagiarism detection in Marathi with a weighted ensemble of TF-IDF and BERT embeddings for low-resource language processing. In *Proceedings of the First Workshop on Language Models for Low-Resource Languages*, 89–100. https://aclanthology.org/2025.loreslm-1.6/

[16] Xiang, L. (2022). Application of an improved TF-IDF method in literary text classification. *Advances in Multimedia*, *2022*(1), 9285324. https://doi.org/10.1155/2022/9285324

[17] Mohammed, M., & Omar, N. (2020). Question classification based on Bloom's taxonomy cognitive domain using modified TF-IDF and word2vec. *PLOS One*, *15*(3), e0230442. https://doi.org/10.1371/journal.pone.0230442

[18] Raharjo, M. M., & Arifin, F. (2023). Machine learning system implementation of education podcast recommendations on Spotify applications using content-based filtering and TF-IDF. *Elinvo (Electronics, Informatics, and Vocational Education)*, *8*(2), 221–230. https://doi.org/10.21831/elinvo.v8i2.58014

[19] Saputro, P. H., Pontoh, F. J., & Tumurang, O. M. (2025). Combination of TF-IDF and Rabin-Karp for detecting document similarity in student thesis abstracts. *Jurnal Teknologi Sistem Informasi dan Sistem Komputer TGD*, *8*(1), 18–24. https://doi.org/10.53513/jsk.v8i1.10611

[20] Zhang, S. (2024). Restaurant recommendation system based on TF-IDF vectorization: Integrating content-based and collaborative filtering approaches. In *Proceedings of the 2023 International Conference on Data Science, Advanced Algorithm and Intelligent Computing*, 610–618. https://doi.org/10.2991/978-94-6463-370-2_62

[21] Senyurek, E., & Kevric, J. (2024). Effects of binary similarity metrics for the relevance component of serendipity-oriented clustering method in collaborative filtering. *IFAC-PapersOnLine*, *58*(9), 172–177. https://doi.org/10.1016/j.ifacol.2024.07.391

[22] Sul, S., & Cho, S.-B. (2024). Understanding people's attitudes in IoT systems using wellness probes and TF-IDF data analysis. *Multimedia Tools and Applications*, *83*(35), 82495–82514. https://doi.org/10.1007/s11042-024-18830-8

[23] Raut, R. M., Kandekar, S., Thorat, P., Jawharkar, G., & Sonawane, S. (2023). Social media content filtering using machine learning techniques. *International Journal of Advances in Engineering and Management*, *5*(2), 794–798.

[24] Zagatti, F. R., Shimizu, G. Y., Lucrédio, D., & Caseli, H. M. (2025). Investigating the relationship between text vectorization cosine similarity and classification performance. *IEEE Access*, *13*, 137348–137363. https://doi.org/10.1109/ACCESS.2025.3595423

[25] Hossain, A. S. M. S., Brown, C., Koop, D., & Malik, T. (2025). Similarity-based assessment of computational reproducibility in Jupyter notebooks. In *Proceedings of the 3rd ACM Conference on Reproducibility and Replicability*, 157–167. https://doi.org/10.1145/3736731.3746159

[26] Jayasundara, H. M. S. M., & Tissera, M. (2025). A bilingual plagiarism detection approach for Sinhala–English text using TF-IDF and cosine similarity. In *2025 International Conference on Advances in Technology and Computing*, 1–6. https://doi.org/10.1109/ICATC68823.2025.11407550

[27] Birthriya, S. K., Ahlawat, P., & Jain, A. K. (2026). Machine learning-based smishing detection using fuzzy logic and TF-IDF feature engineering. *Franklin Open*, *14*, 100506. https://doi.org/10.1016/j.fraope.2026.100506

How to Cite: Muindisi, G. T., Nuri, Y., & Senyurek, E. (2026). A Comparative Analysis of Two Algorithms for TF-IDF-Based Document Similarity. *Journal of Data Science and Intelligent Systems*. https://doi.org/10.47852/bonviewJDSIS62027582