WWW.IJRPR.COM

International Journal of Research Publication and Reviews

Journal homepage: www.ijrpr.com ISSN 2582-7421

Plagiarism Analyzer And Detector

Prof Swathi .A[1], Varshini A[2], Sri Vidhya MJ[3], Tejaswini C[4], Soujanya Ratnakar Naik[5]

[1] Dayananda Sagar Academy of Technology and Management swathianjanappa7777@gmail.com
[2] Dayananda Sagar Academy of Technology and Management varshini14204@gmail.com
[3] Dayananda Sagar Academy of Technology and Management srividhya044@gmail.com
[4] Dayananda Sagar Academy of Technology and Management tejaswininayaka077@gmail.com
[5] Dayananda Sagar Academy of Technology and Management soujanyanaik903@gmail.com

ABSTRACT :

This AI-based platform makes for the integration of advanced review features with plagiarism detection, capable of streamlining content verification for end- users. The basis is built on natural language processing and machine learning algorithms, ensuring accurately highlighted duplicated or paraphrased content across multiple sources, along with a recommendation list. It provides for rich reports with similarity indexes and source matches.

Beyond detecting plagiarism, the web-based tool includes an inbuilt review feature. This inbuilt review system lets users appraise and score content that fits a particular set of parameters like originality, coherence, and relevance. With such features, dual functionality produces an environment which fosters robust academic and professional use with creative development towards the generation of true, authentic content of higher quality. Integrating both automation and user-friendly interfaces ensures better productivity along with an ethical approach toward content and content-related tasks in each respective field.

Index Terms – Plagiarism Detection, Cosine Similarity, Vector Space Models, Natural Language Processing, Text Analysis, Term Frequency–Inverse Document Frequency [TF-IDF].

Introduction :

Plagiarism is any unauthorized use of components or the full of any work without giving proper credit to the initial creator hence it takes away the originality. Plagiarism is also very common in today's computer science education as one can easily duplicate and adapt digital assignments. Students continue plagiarism even under the threat of a punishment.

Besides this, students are innovative when it comes to hiding their trace of plagiarism to avoid pointing to the origin. In assignments that require submitting code, the common methods adopted by them are renaming, re- ordering or restructuring . This definition includes all types of creative works, irrespective of their status as published or unpublished, including manuscripts, printed materials, and digital formats.

Plagiarism can be deliberate, careless, or accidental. Intentional or negligent plagiarism is strictly prohibited under examination rules, and disciplinary actions may be taken as a result. Plagiarism is bad because, first and foremost, it is wrong to claim ownership of another person's work. Essentially, plagiarism is a form of theft. Second, plagiarism is against the rules of academic integrity. Colleges teach students how to share material responsibly and give creators proper credit. Plagiarism contravenes these laws.

Plagiarism also devalues college degrees. Everyone loses if businesses believe that many students plagiarized to graduate. Although academic plagiarism is not a crime, it does violate institutions' academic honesty policies. Consequently, plagiarism can have severe consequences for students, including expulsion or academic probation. AI-based plagiarism detection software helps tutors and teachers identify cases of plagiarism in students' work by using scanning and coding programs. However, our initial approach is vulnerable to re-ordering attacks and is also not widely applicable as it can only be applied for metamodels. Moreover, it is also unclear how well both approaches will

perform against the growing menace of AI-based plagiarism. In this paper, we therefore present a plagiarism detection approach for modeling tasks that surpasses the state-of-the-art techniques. It also helps in detecting the plagiarized content in either way, it might be paraphrased or reordered or any other alterations has been done to the original copy.

Literature survey :

Plagiarism detection has come a long way, evolving from simple string-matching techniques to sophisticated machine learning models that capture deeper semantic similarities between texts. With the continuous expansion of available digital content, the field will likely continue evolving, especially in areas like paraphrase detection, cross-language plagiarism, and domain-specific applications such as software plagiarism. Tools and methodologies will need to adapt to new forms of plagiarism and the challenges posed by evolving digital ecosystems. While tools like Turnitin, MOSS, and PlagScan are widely used, ongoing research focuses on improving accuracy, especially for paraphrasing, cross- language plagiarism, and plagiarism in multimedia.

Cosine Similarity

Cosine similarity is probably one of the widely practiced techniques of detecting the similarity in texts or documents. Cosine of the angle existing between two n- dimensional space vectors helps one to evaluate the similarities that occur in two given documents. In that process, the text was tokenized, stopwords are removed and terms are converted into TF-IDF vector representation. The range for the cosine similarity score, as provided by the cosine similarity between documents, is between 0 to 1. The closer value to 1, therefore, indicates a higher closeness between two documents. Because of its efficiency and ability in large sets to detect local similarity, researchers prefer this method of using vector space models.

Vector Space Model

The VSM extends cosine similarity by representing documents as vectors in a multi-dimensional space. Each dimension corresponds to a unique term in the corpus. VSM enables comparison of documents by quantifying similarity using measures like Euclidean distance or cosine similarity. It is effective for large-scale text analysis because of its mathematical rigor and flexibility in incorporating weightings such as TF-IDF. In plagiarism detection, the VSM is used to compare textual contents for near-duplicate and paraphrased sections. Research shows it to be effective in identifying both verbatim and conceptual plagiarism across various document collections.

File Similarity Techniques

File similarity detection aims at the analysis of the overall structure and content of files for similarity. Hash-based comparisons, n-gram analysis, and fingerprinting are common methods used. These approaches divide the text into subcomponents like substrings or groups of words, and calculate the similarity score on the basis of overlapping these components. N-gram analysis, for example, is useful for identifying text that has been reworded. Fingerprinting, by contrast, can better work with document structures to be compared. File similarity techniques perform very well in hybrid models by integrating it with cosine similarity and VSM.
The integration of cosine similarity, VSM, and file similarity techniques provides a comprehensive framework for plagiarism detection. These methods complement each other by addressing various aspects of text and file analysis, ensuring accurate and reliable results. Future research may focus on enhancing these techniques with advancements in natural language processing (NLP) and machine learning, enabling more nuanced and context- aware detection mechanisms.

Methodology :

The architecture of the system includes the following steps:

Preprocessing

Preprocessing is one of the primary steps in standardizing and cleaning input texts in order to acquire accurate features of meaningful extraction. Generally, input texts are tokenized into individual words or terms that break down the text into manageable components. To remove noise, NLTK removes stopwords such as "and," "the," using a list that is very large for each language. Those words are frequent but do not contribute much to the semantic content of the text. Punctuation marks are also removed so that those are not affecting the vector representations. Lastly, all text is transformed into lowercase to ensure uniformity and not to suffer from case sensitivity while vectorizing.

Text Vectorization

The texts are converted into vector representations using:

Cosine Similarity: It is the cosine of the angle between two non-zero vectors. It is given by:

$$\text{cosine similarity} = S_C(A,B) := \cos(\theta) = \frac{\mathbf{A}\cdot\mathbf{B}}{\|\mathbf{A}\|\|\mathbf{B}\|} = \frac{\sum_{i=1}^{n} A_i B_i}{\sqrt{\sum_{i=1}^{n} A_i^2}\cdot\sqrt{\sum_{i=1}^{n} B_i^2}},$$

[8]

Here, (A and B) are term frequencies in the query and reference texts, respectively.

2.2.3 TF-IDF Enhanced Representation: It improves feature representation by reducing the weight of frequently occurring but uninformative terms. The Term Frequency- Inverse Document Frequency (TF-IDF) method improves
vectorization by dampening words that appear frequently in all documents but do not contribute much to the semantics; for example, "is," "was." More importantly, TF- IDF gives greater weights to words that are distinctive or relevant to the document. Thus, the quality of feature representation is improved

Similarity Computation

Two techniques are combined:

Cosine Similarity Analysis (Algorithm 1): This algorithm computes similarity directly on text vectors. The algorithm measures the weighted alignment of terms that exist between two vectors to identify similarities based on directional closeness.

File Similarity Matching (Algorithm 2): It is an elaboration of the simple cosine similarity technique using term frequency-based scoring mechanisms. It is checking the overall similarity in input text with a database of documents, in terms of term weights and their frequencies for the comprehensive percentage match.

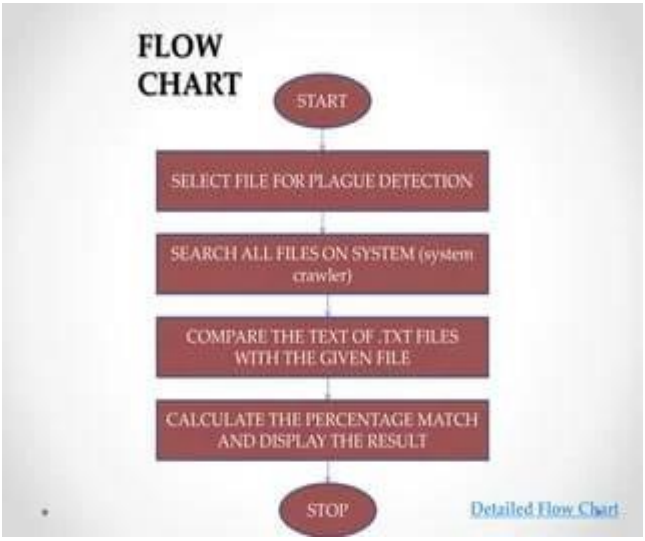

Figure 1: Flowchart of how the AI website works

Implementation :

Cosine Similarity Function

The cosineSim function computes the cosine similarity of two strings. It first converts them into vectors by using the text_to_vector function. It uses intersection of vectors to compute weighted similarity.

File Similarity Function

The findFileSimilarity function integrates stopword removal and frequency analysis to compute similarity percentages.

Code Example

The following Python code demonstrates the implementation:

```
def cosineSim(text1, text2):
vector1 = text_to_vector(text1.lower())
vector2 = text_to_vector(text2.lower())
return get_cosine(vector1, vector2)
def findFileSimilarity(inputQuery, database):
dotProduct = dotProduct(vector1, vector2)
matchPercentage = (dotProduct / (queryVectorMagnitude
* databaseVectorMagnitude)) * 100 return matchPercentage.
```

Conclusion :

In conclusion this paper introduces an efficient hybrid approach to plagiarism detection, leveraging NLP and mathematical techniques. The system exhibits improved accuracy for various similarity scenarios, making it a valuable resource for researchers, educators, and publishers. Future work will explore integrating deep learning models for enhanced semantic understanding. The study demonstrates the effectiveness of combining Cosine Similarity with term frequency analysis for plagiarism detection. The results highlight the complementary strengths of Cosine Similarity and VSM. While Cosine Similarity excels at capturing structural similarities, VSM accounts for semantic nuances, making the hybrid approach robust against various plagiarism

tactics. By incorporating stopwords removal and vector-based computations, the system can accurately identify overlaps while reducing false positives. Limitations include dependency on preprocessing quality and challenges with highly technical or jargon-rich texts.

REFERENCES :

1. Han, J., Kamber, M., & Pei, J. (2011). Data Mining:Concepts and Techniques. Morgan Kaufmann.
2. Salton, G., Wong, A., & Yang, C. S. (1975). A vector space model for automatic indexing. Communications of the ACM, 18(11), 613-620.
3. Luhn, H. P. (1957). A statistical approach to mechanized encoding and searching of literary information. IBM Journal of Research and Development, 1(4), 309-317.
4. Jurafsky, D., & Martin, J. H. (2023). Speech and Language Processing (3rd ed.). Pearson.
5. Ramos, J. (2003). Using TF-IDF to determine word relevance in document queries. https://doi.org/10.17700/jai.2019.10.1.426
6. Potthast, M., Stein, B., & Anderka, M. (2010).
7. A Wikipedia-Based Multilingual Retrieval Model for Plagiarism Detection. Alzahrani, S. M., Salim, N., & Abraham, A. (2012). Understanding Plagiarism Detection Techniques Based on Machine Learning. Expert Systems with Applications, 39(1), 1339-1356.
8. Manning, C. D., Raghavan, P., & Schütze, H. (2008). Introduction to Information Retrieval. Cambridge University Press.
9. Chowdhury, G. G., & Lehal, G. S. (2007) Plagiarism detection tools and techniques. Journal of the American Society for Information Science and Technology, 58(2), 249–267.
10. Stein, B., Lipka, N., & Prettenhofer, P. (2011) Intrinsic Plagiarism Analysis. Language Resources and Evaluation, 45(1), 63–82.
11. Foltynek, T., Meuschke, N., & Gipp, B. (2019) Academic Plagiarism Detection: A Systematic Literature Review. ACM Computing Surveys (CSUR), 52(6), 1-42.
12. Barrón-Cedeño, A., & Rosso, P. (2009) On Automatic Plagiarism Detection Based on n-Grams Comparison. European Conference on Information Retrieval (ECIR), 696–700.