Combining Approximate String Matching Algorithms and Term Frequency In The Detection of Plagiarism

Zina Balani *zina.0174810@gmail.com*
Department of Software Engineering
Koya University
Koy sinjaq, 44023, Iraq

Cihan Varol *cvarol@gmail.com*
Department of Computer Science
Sam Houston State University
Huntsville, TX 77341, USA

Abstract

One of the key factors behind plagiarism is the availability of a large amount of data and information on the internet that can be accessed rapidly. This increases the risk of academic fraud and intellectual property theft. As increasing anxiety over plagiarism grow, more observation was drawn towards automatic plagiarism detection. Hybrid algorithms are regarded as one of the most prospective ways to detect similarity of everyday language or source code written by a student. This study investigates the applicability and success of combining both the Levenshtein edit distance approximate string matching algorithm and the term frequency inverse document frequency (TF-IDF), thereby boosting the rate of similarity measured using cosine similarity. The proposed hybrid algorithm is also able to detect plagiarism occurred on natural language, source codes, exact, and disguised words. The developed algorithm can detect rearranged words, inter-textual similarity of insertion or deletion and grammatical changes. In this research three various dataset are used for testing: automated machine paragraphs, mistyped words and java source codes. Overall, the system proved to be detecting plagiarism better than the yet alone TF-IDF approach.

Keywords: Approximate, Hybrid, Plagiarism, Similarity, TFIDF.

1. INTRODUCTION

Onward entering the age of digital communication, the ease of sharing information across the internet has facilitated online literature searches. This carries the confidential risk of increased academic misconduct and educated attribute theft, as concerns about plagiarism increase [1]. Plagiarism is regarded a serious problem in modern research manuscripts [2]. This work will focus on two plagiarism areas: source code (also called "code clone") and natural language. Plagiarism in natural language is relatively hard to detect due to the rich morphological, syntactic, and semantic properties of natural language. Moreover, plagiarists use various paths to overcome plagiarism detection systems by replacing the original text using various types of hiding and using smart plagiarism techniques, including rewriting, synonym substitution, paraphrasing, text processing, text translation, and idea adoption [3]. To mitigate the negative effects of plagiarism, systems aim to identify cases of theft in paper or documents via comparing a large collection of documents with suspicious documents. Finally, systems detect if the suspicious document has been stolen or similar to each other [4].

There are two essential methods for automatic plagiarism detection: extrinsic / external plagiarism detection and intrinsic / internal plagiarism detection. Intrinsic plagiarism detection only analyzes the input document and finds some parts that have not been created by the same author without performing a comparison with an external corpus. External plagiarism detection requires a

reference collection of documents that are considered as original. Suspicious documents are compared to all documents in this collection to detect duplicate or approximately matching components in the source document [5]. This study deals with external plagiarism detection methods.

Strings can be identical in two manners. There are cases where they share syntactically the same character sequence and cases where they have the same semantic meaning (such as synonyms). Similarity measures can break into four classes: character-based, q-gram, token-based, and mixed measures. Character-based and q-gram measures calculate similarity based on a sequence of characters in two strings. Token-based measures use white space, line breaks, or punctuation characters to break a string into words and symbols (called tokens), and calculate the similarity between two token sets. Mixed measures are a combination of character-based measures and token-based measures [6].

At the beginning of the 20th century, in the early 1970s, scientists have studied a large-scale copy of the program to prevent technology and software. There are several classic systems available for detecting plagiarism in program source code. Plagiarized program code is different than the plain language as different codes can perform the same function. Some smart plagiarists use certain methods to alter the code. For example, for loop becomes a while loop or adding large number of randomly generated intermediate variables [7].

In today's developed technology where techniques of plagiarism detection are available, previous methods and techniques that were used to detect and find measures of plagiarism are no longer in use. Most investigators are now working on the success of hybrid algorithms to find the similarity.

Therefore, in this study the applicability and success of two specific hypothesis are investigated:

• Approximate string matching algorithms can be employed to detect plagiarism.
• Combining approximate string matching with TF-IDF will increase the plagiarism detection rate.

The proposed system can identify different types of plagiarism such as sentence reordering, source code, inter-textual similarity, and approximate copy similarity. Edit distance is used to find the similarity for mistyped textual information, if the obtained similarity measure exceeds the predefined threshold which is greater than or equal to 0.50. Once this threshold is reached, then TF-IDF counts the word as its original, which increases the frequency of the common words. As a result, the rate of plagiarism will be boosted which is measured by Cosine Similarity. The success rate of a hybrid version of approximate string matching and term frequency is investigated, and is the results reflected that a robust hybrid algorithm is capable to detect both plagiarism on natural language and source code (Code Clone).

2. LITERATURE REVIEW

A wide range of problems can be emerged when many students copy someone else's work and submit it as their own. To detect these cheats, the online learning management systems started to add plagiarism detection tools, and when two identical or sufficiently similar assignments are detected a flag is raised for the issue. However, plagiarists use a variety of methods to alter the submitted work to avoid detection by the system. In recent years, the problem of detecting plagiarism in natural language has attracted the attention of many researchers. A number of plagiarism detection tools have been developed specifically for English language.

According to [1], one of the oldest methods of detecting plagiarism was introduced by Bird in 1927, who investigated the application of statistical methods to detect plagiarism in multiple choice answers. Afterwards, methods developed in the 1960s pointed on detecting plagiarism in multiple choice tests. Levenshtein Distance (LD) was renamed after Vladimir Levenshtein, a Russian scientist who designed the algorithm in 1965, and is also known as edit distance. It's a

measure of the similarity between two strings called source string (s) and target string (t). Distance is the number of deletions, additions, or replacements required to convert s to t [8]. This technique was used for near duplicate detection for several years. Back in the 1970s, researchers began working on similarity detection techniques for source code [9]. Since 1970s, numerous tools have been introduced to measure the similarity of code. They are used to address issues like code duplication detection, software license violations, and software plagiarism [10].

In 1975, Halstead suggested the first algorithm called the property counting method. The algorithm computed the operators and operands statistics seeming in the source program and used it as the main basis for determining the result [9]. Between the 1970s till mid-1990s was a golden period of developing different methods and algorithms of exact and approximate string matching algorithms [11]. Since the mid-1990s, the focus of research has shifted to the study of natural language text. In 1994 Manber suggested the concept of an approximate fingerprint. The basic rule is to measure the similarity among documents by string matching. This principle has been accepted by most researchers. Therefore, based on this, researchers increased word frequency counting, keyword extraction techniques, and accomplished matching by computing hash values for the text [9]. Experimental outcomes indicates that a hybrid technique that considers word frequency and character-based word similarity increases matching. The first venture in this direction was found in 2003 by Cohen et al. A measurement format named Soft-TFIDF, which extends the Jaro-Winkler method to combine the frequency weight of a word in a measure of cosine similarity and a measure of CLOSE at the character level [12]. In 2007, according to [1], Dreher proposed using a normalized word vector algorithm to calculate similarity based on VSM (Vector Space Model), which performs synonym generalization for each word. Despite the advent of more advanced approaches, paraphrase remained difficult. In 2012, Ekbal, Saha, and Choudhary suggested ways to detect plagiarism. This includes three main steps. First, the basic tasks of natural language processing are performed in preprocessing steps. The second step selects a set of source documents similar to the suspect document. A VSM is used to identify the source document for each suspect document. Finally, the similarity between the two documents is calculated using the cosine similarity. If the resultant measure of similarity exceeds a predefined threshold, the document is considered as a source document for the suspect document. In the third step, similar text is found in both source and suspect documents using the n-gram method [13]. In 2013, Investigators Combined both of VSM and Jaccard coefficient into one, the method fully utilizes the benefits of VSM and Jaccard coefficient, and it can extract more reasonable heuristic seeds in plagiarism detection. The preliminary outcomes indicate that the method can generate better performance. [14]. In 2018, a system was recommended for Urdu text, based on a distance measurement method, structural alignment algorithm, and vector space model. System performance is measured using machine learning classifiers (Support Vector Machine and Naive Bayes). Experimental outcomes demonstrate that the output of the suggested method is advanced compared to other existing model (i.e. cosine method, simple Jaccard measure) [15]. According to [16], the hybrid of the Rabin-Karp and Levenshtein algorithms provides a useful contribution. The degree of identity can be optimized so that documents can be classified precisely according to the content of the document. The hybrid algorithm is able to enhance the precision of the evaluators according to certain parameters, N-Gram, Base and Modulo. In 2021, AL-Jibory proposed an external plagiarism detection strategy based on a system integrating natural language processing (NLP) and machine learning (ML) techniques, as well as text mining and similarity analysis. The proposed technique uses a combination of Jaccard and Cosine similarity demonstrated by a design application used to identify plagiarism in scientific publications. [17]

The literature declared that till now the matching algorithms for detecting plagiarism are challenges. Therefore, the hybrid algorithms are advanced. Combining of Levenshtein Edit Distance LED and Term Frequency-Inverse Document Frequency TF-IDF behind of detecting plagiarized words, is also capable to enhance the rate of the plagiarism by detecting disguised words. Meantime, term frequency is implemented which is a powerful techniques can also detect the plagiarized source code from different programming languages.

3. METHODOLOGY

3.1 Edit Distance and TF-IDF Based Algorithms

• Levenshtein Edit Distance: One of the effective methods of comparing strings, two terms or mainly two sequences is the edit distance which calculates the cost of the optimal sequence of the editing process by adding, removing and replacing. There are multiple differences in the computation of the edit distance, while the Levenshtein distance proposed in 1966 being the most known in the literature. Distance is the minimum number of machining process that are converted from one string to another. Allowed editing operations are the adding, removing and substituting of individual characters [4]. Small discrepancies in the input data can still be pointing a plagiarism. This is why approximate string matching approach is used to cover this challenging area. A threshold between suspicious papers and the source document repository is established. If the predefined threshold is greater than or equal to 0.50, the word is considered similar to the original. Later, TF-IDF, which is explained below, is used to determine the frequency of the words to have a better understanding of the plagiarism rate.

• TF-IDF Weighting: TF-IDF is performed as an important determinant in text mining and information retrieval, which enables the establishment of a vector space where every vector indicates how a word is significant for a document in an aggregation through the combination of Term Frequency TF (t, d) and Inverse Document Frequency IDF (w) [18]:

a) Term Frequency TF it refers to the number of times a word is included in a document. Since the length of each document is various, a term may appear much more in long documents than in short documents. TF is defined as:

$$TF_{(t,d)} = \frac{O}{t}$$

where O refers the amount of times that a term t occurs in a document, and t is the number of words in the paper.

b) Inverse Document Frequency IDF is a statistical weight performed to measure the significance of a word in a collection of text documents. It also has a built-in IDF functionality that decreases the weight of terms that appear regularly in the document set and increases the weight of words that appear infrequently. IDF is defined as:

$$IDF_{(t,d)log} = \frac{|D|}{N}$$

Where |D| is the complete number of documents and N is the number of documents with the term t.

c) Term Frequency-Inverse Document Frequency TF-IDF is computed for every word in the article by combining TF and IDF.

$$TF - IDF(t, d, f) = TF(t, d) * IDF(t, d)$$

TF-IDF algorithm is performed to detect the relevance to a query document and TF-IDF weighting for representing the percentage of terms in a document. Terms with high of TFIDF weighting indicates a powerful connection to the document, including the document query.

3.2 Vector Space Model with Cosine Similarity Measure

The vector space model (VSM) is one of the common techniques that uses the lexical and syntactic characteristics and describes the article in vector space. Several weighting schemes are then used for document presentations and comparisons. Term-frequency-inverse document frequency (TF-IDF) and term frequency-inverse sentence frequency (TF-ISF) are mainly used as two weighting schemes [19]. The TF-IDF weighting technique is frequently used in the vector space model along with similarity methods to find the similarity among two articles. Common measurements based on the vector space model are cosine similarity and jaccard coefficient,

performed to represent text papers (usually every objects) as vectors in multidimensional space. In this study cosine similarity measure is applied.

The cosine similarity is a measure of the similarity that is computed by multiplying the cosine angles of the two vectors to be compared. The cosine 0 ° is 1, which is less than 1 to the value at any other angle. Thus, the similarity values of the two vectors are: If the cosine similarity value is 1, they are similar. This method is a traditional method that is often used and combined with the TF-IDF. The calculation of cosine similarity is implemented according to the below equation [20]:

$$Cos\alpha = \frac{A * B}{|A| * |B|} = \frac{\sum_{i=1}^{n} A_i * B_i}{\sqrt{\sum_{i=1}^{n} (A_i)^2} * \sqrt{\sum_{i=1}^{n} (B_i)^2}}$$

where A is the weight of every attribute of vectors A and B is the weight of every attribute in vector B.

Subsequently generating the vectors and converting the attributes in the vectors to weighted values, we can use cosine similarity measure to compute the likeness of those vectors. Basics of cosine similarity, the bigger angle shaped among two coordinate vector comparison papers, lower degree of similarity of papers. Moreover, the smaller degree of cosine similarity level means the similarity rate will be bigger [20].

3.3 System Architecture

The proposed system performs the following steps which are represented in figure 1.

1. Levenshtein Edit Distance Algorithm is implemented to find near identical information between the source and suspicious document. If determined threshold is greater or equal to 0.50 then the word is considered as plagiarized. Later, TF-IDF counts the word as it is original, which increases the rate of detecting plagiarism. Otherwise, the word considered as not plagiarized.
2. Cosine Similarity used as an efficient way to determine the similarity measures.

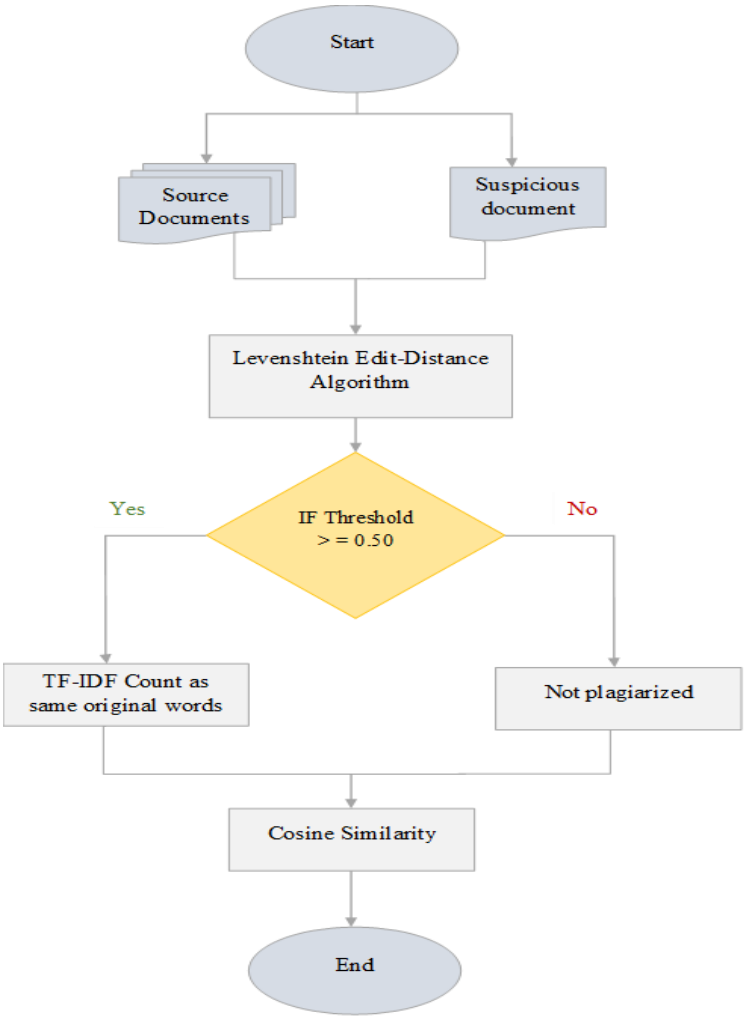

FIGURE 1: Proposed system flowchart for plagiarism detection system.

3.4 Implementation

Two techniques of matching algorithms have been successfully implemented to enhance plagiarism rate. Initially Levenshtein Edit Distance LED based on approximate string matching algorithms and Term Frequency Inverse Document Frequency TF-IDF based on exact string matching algorithms. Following, after finding Vector Space Model VSM, cosine similarity is implemented to determine the percentage of similarity between documents. Java programming language has been implemented in this project. And eclipse is used as a tool to execute the codes.

4. EXPERIMENTS AND RESULTS

The advanced plagiarism detection system is based on two similarity metrics. Accordingly, a percentage threshold is usually predefined in order to determine the cases of plagiarism to be found. Levenshtein edit distance has been implemented to determine near-identical similarity measure. If the percentage of plagiarism detected is greater or equal to 0.50 the system assumes that there is plagiarism and TF-IDF counts as it is a textual information identical to the one in original, which helps to identify near identical plagiarized information. Later, Cosine similarity measure is used to find the distance between the original and suspect document. Moreover, if the threshold is smaller than 0.50, the plagiarism will not be found. The outcomes demonstrates that the hybrid algorithm is more powerful than others. Because it can detect disguised or mistyped

words in both of natural language and source codes and counts as its original which will boosts the rate of the plagiarism.

4.1 Datasets

This research contributes in providing a dataset for natural English language and source code plagiarism detection. The corpus contains several types of plagiarism cases including: simple copy/paste, word shuffling, and phrase shuffling, removed character, added diacritics, and paraphrasing. The corpus is expected to perform plagiarism detection approaches of available source documents comparable to each other. Plagiarism text files specific to this dataset are created for plagiarism purposes.

Three different datasets are used for testing: First, automated machine paragraphs [21] has the dataset size of 60.8MB which contains 79,970 documents, and it is divided into two sets: source documents (39,241) and suspicious document (40,729). Second dataset is Java source codes which were obtained from Githubs GH and stack over flows SO [22]. The dataset contains 180 java files, 97 of them are Github GH source code and 83 of them are on Stack Overflow SO. Finally, a Misspelled dataset [23] contains more than 6,000 misspelled words was used for testing. Table 1 demonstrate the result of our research after simulation test the optimum threshold is found to be 0.5.

No	Algorithms	Plagiarism Rate
1	Edit Distance	56.4%
2	TF-IDF	27.9
3	The Hybrid Algorithm	74.1%

TABLE 1: Comparison of plagiarism rate for algorithms .

4.2 Evaluation Measures

In order to evaluate the success of the result, three evaluation criteria are implemented in this study: precision and recall are two values that cooperatively are applied to assess the efficiency of performance of information retrieval systems. The F1 score is also a useful indicator for comparing two classifiers. F1 score produced by determining the harmonic mean of precision and recall. The results show that the version of the hybrid algorithm of approximate string-matching algorithms and term frequency is the most effective one which has a 0.851 F1 score, edit distance which has a score of 0.721 and TF-IDF is 0.436.

5. CONCLUSION AND FUTURE WORK

Edit distance provides an indication of similarity that may be too close in some cases. If user duplicates java code and performs a several alternates, such as, change of variable names, addition of two comments, the edit distance between the source and copy will be close. On the other hand, TF-IDF is the most commonly used weighting scheme for keywords to simplify all related articles. However, the current TF-IDF technique does not take into account the semantic correlations between terms, which can lead to a less relevant search for documents. Therefore, this study combined both the Levenshtein edit distance based approximate string matching algorithm and the term frequency inverse document frequency TF-IDF, thereby increases the score of similarity measured using cosine similarity. The robust hybrid algorithm is also able to detect plagiarism from both the natural language and source codes. Different forms of plagiarism can be detected such as (reorganization of words and inter-textual similarity of insertion / deletion). It can detect different types of plagiarism, such as added comments in Java source code or changes in the data fields and methods etc. In this investigation three different corpus are used for testing: automated machine paragraphs, mistyped words, and java source codes. From the results we find that the proposed algorithms are capable of detecting disguised and exact copy of articles which boosts the rate of the plagiarism detection.

In the future, we plan to extend the hybrid algorithm to be able to detect paraphrased text. We also plan to expand the scope of the source dataset with advanced programming topics (e.g. finding and sorting) and code files from other programming courses (e.g. object-oriented programming or algorithms and data structures).

6. REFERENCES

[1] M. Y. M. Chong, "A study on plagiarism detection and plagiarism direction identification using natural language processing techniques," 2013.

[2] S. Rani and J. Singh, "Enhancing Levenshtein's edit distance algorithm for evaluating document similarity," in International Conference on Computing, Analytics and Networks, 2017: Springer, pp. 72-80.

[3] H. Cherroun and A. Alshehri, "Disguised plagiarism detection in Arabic text documents," in 2018 2nd International Conference on Natural Language and Speech Processing (ICNLSP), 2018: IEEE, pp. 1-6.

[4] A. Zouhir, R. El Ayachi, and M. Biniz, "A comparative Plagiarism Detection System methods between sentences," in Journal of Physics: Conference Series. Vol. 1743. No. 1. IOP Publishing, 2021

[5] R. R. Naik, M. B. Landge, and C. N. Mahender, "A review on plagiarism detection tools," International Journal of Computer Applications, vol. 125, no. 11, 2015.

[6] N. Gali, R. Mariescu-Istodor, and P. Fränti, "Similarity measures for title matching," in 2016 23rd International Conference on Pattern Recognition (ICPR), 2016: IEEE, pp. 1548-1553.

[7] L. Qinqin and Z. Chunhai, "Research on algorithm of program code similarity detection," in 2017 International Conference on Computer Systems, Electronics and Control (ICCSEC), 2017: IEEE, pp. 1289-1292.

[8] X. Wang, S. Ju, and S. Wu, "Challenges in Chinese text similarity research," in 2008 International Symposiums on Information Processing, 2008: IEEE, pp. 297-302.

[9] X. Liu, C. Xu, and B. Ouyang, "Plagiarism detection algorithm for source code in computer science education," International Journal of Distance Education Technologies (IJDET), vol. 13, no. 4, pp. 29-39, 2015.

[10] C. Ragkhitwetsagul, J. Krinke, and D. Clark, "A comparison of code similarity analysers," Empirical Software Engineering, vol. 23, no. 4, pp. 2464-2519, 2018.

[11] K. Al-Khamaiseh and S. ALShagarin, "A survey of string matching algorithms," Int. J. Eng. Res. Appl, vol. 4, no. 7, pp. 144-156, 2014.

[12] T. El-Shishtawy, "A hybrid algorithm for matching arabic names," arXiv preprint arXiv:1309.5657, 2013.

[13] A. Ekbal, S. Saha, and G. Choudhary, "Plagiarism detection in text using vector space model," in 2012 12th International Conference on Hybrid Intelligent Systems (HIS), 2012: IEEE, pp. 366-371.

[14] S. Wang, H. Qi, L. Kong, and C. Nu, "Combination of VSM and Jaccard coefficient for external plagiarism detection," in 2013 international conference on machine learning and cybernetics, 2013, vol. 4: IEEE, pp. 1880-1885.

[15] W. Ali, Z. Rehman, A. U. Rehman, and M. Slaman, "Detection of plagiarism in Urdu text documents," in 2018 14th International Conference on Emerging Technologies (ICET), 2018: IEEE, pp. 1-6.

[16] A. H. Lubis, A. Ikhwan, and P. L. E. Kan, "Combination of levenshtein distance and rabin-karp to improve the accuracy of document equivalence level," International Journal of Engineering & Technology, vol. 7, no. 2.27, pp. 17-21, 2018.

[17] F. K. AL-Jibory, "Hybrid System for Plagiarism Detection on A Scientific Paper," Turkish Journal of Computer and Mathematics Education (TURCOMAT), vol. 12, no. 13, pp. 5707-5719, 2021.

[18] A. Mahmoud and M. Zrigui, "Semantic similarity analysis for paraphrase identification in Arabic texts," in Proceedings of the 31st Pacific Asia conference on language, information and computation, 2017, pp. 274-281.

[19] D. Gupta, "Study on Extrinsic Text Plagiarism Detection Techniques and Tools," Journal of Engineering Science & Technology Review, vol. 9, no. 5, 2016.

[20] A. R. Lahitani, A. E. Permanasari, and N. A. Setiawan, "Cosine similarity to determine similarity measure: Study case in online essay assessment," in 2016 4th International Conference on Cyber and IT Service Management, 2016: IEEE, pp. 1-6.

[21] Foltynek, T., Ruas, T., Scharpf, P., Meuschke, N., Schubotz, M., Grosky, W., Gipp, B. Detecting Machine-obfuscated Plagiarism [Data set], University of Michigan - Deep Blue, 2019.

[22] Baltes, Sebastian. Usage and Attribution of Stack Overflow Code Snippets in GitHub Projects — Supplementary Material (Version 2017-01-15) [Data set], 2018.

[23] Wahle, Jan Philip, Ruas, Terry, Foltynek, Tomas, Meuschke, Norman, & Gipp, Bela. Identifying Machine-Paraphrased Plagiarism (Version 1.0) [Data set], 2021.